
.

# TREBALL FI DE CARRERA

**TÍTOL DEL TFC: Aplicació web per la gestió de incidències.**

**TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica**

**AUTOR: Agustí Grau**

**DIRECTOR: Miguel Valero**

**SUPERVISOR: Pablo Hontoria**

**DATA: 26 de Febrer de 2008**

**TÍTOL DEL TFC: Aplicació web per la gestió de incidències.**

**TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica**

**AUTOR: Agustí Grau**

**DIRECTOR: Miguel Valero**

**SUPERVISOR: Pablo Hontoria**

**DATA 26 de febrer de 2008**

Aquesta memòria descriu el procés de desenvolupament d'un aplicatiu de gestió de les incidències i avaries d'una WAN, per part d'un servei de Helpdesk.

La plataforma d'aquest aplicatiu és BEA Weblogic i Oracle 8.1, proporcionant un aplicatiu web robust. Aquest aplicatiu Web permet la introducció, consulta, modificació i eliminació de les incidències. Aquest software permet fer el seguiment en funció de diversos criteris, realitza informes cronològics i representacions gràfiques.

Quan s'introdueix una incidència, el seguiment comença amb la generació i enviament automàtic d'un informe al usuari per tenir coneixement de l'estat de la seva consulta. El software fa un seguiment de la incidència des de la comunicació fins que l'usuari final la considera finalitzada.

El punt inicial d'aquest projecte és l'antic software, i les millores i funcionalitats que són requerides pel client. El disseny i la implementació són fetes en funció de les necessitats del client i de l'entorn de desenvolupament de la nova plataforma.

Per últim, aquesta memòria descriu les proves fetes a l'inici del temps de desenvolupament pel programador i el client, a traves de visites, i finalitzades amb la instal·lació i la prova funcional final. Les conclusions i les millores que poden ser fetes al software són recollides per esdevenir la part final del treball.

**TITLE : Development of an application of incidence management**

**AUTHOR : Agustí Grau**

**DIRECTOR : Miguel Valero**

**SUPERVISOR : Pablo Hontoria**

**DATE : 26th Febrer 2008**

This report describes the development of an alarms and faults management software for a WAN, as a part of helpdesk service.

The platform of development of the software is BEA Weblogic 9.2 and Oracle 8.1, providing a strong Web application. This is a Web application that manages the entering, search, modification and deletion of the alarms. The software allows the tracking depending on the criteria of the alarms, generates chronological reports and graphical representations.

When an alarm is introduced, the tracking begins with a report that is generated and sent automatically informs the user of the status of his alarm. The software makes a tracking of the incidence since the beginning of communication until the final user considers it closed.

The starting point of this project is former software, and the improvements and functionalities that are needed. The design and implementation are done to satisfy the customer needs and the development environment of the new platform.

Finally, this report describes the tests that started during the development time, by the developer and the customer, through visits, that ended with the installation and the final functional test. The conclusions and the improvements that can be done to this software are collected as the final part of this thesis.

# ÍNDEX

# 1 INTRODUCCIÓ

Les xarxes informàtiques cada cop són més necessàries en qualsevol organització que vulgui tenir la capacitat de gestionar la informació tant de forma interna com a canal de comunicació amb els seus clients, proveïdors, etc. d'una manera ràpida i eficient. En les grans organitzacions la estabilitat de la xarxa es fonamental per al funcionament de la pròpia empresa. Les xarxes informàtiques estan formades per els nodes de interconnexió, els elements de connexió i els equips informàtics i la bona gestió de cadascun d'aquests element implica el bon funcionament del conjunt de la xarxa. Òbviament, cadascun d'aquests elements necessitarà d'una atenció diferenciada, depenent de la funció exacta de cadascun d'ells. En el cas del equips informàtics, aquesta gestió implica l'assessorament dels usuaris finals en l'ús d'aquests: manteniment del programari i la configuració del mateix, a més de la resolució de qualsevol tipus d'incidència tècnica que pugui ocórrer en els equips. El servei dintre d'una organització que s'encarrega d'aquestes tasques rep el nom de Helpdesk, o ajut a escriptori, en referència a que el destinatari és l'usuari final.

Aquest projecte explica el disseny i desenvolupament d'un aplicatiu destinat a la gestió d'incidències dintre una xarxa informàtica de tipus corporatiu, a partir d'una aplicació existent, a més, el desenvolupament d'aquest aplicatiu s'ha fet dintre de l'empresa ADTEL Sistemas de Telecomunicación, SL,. Aquesta aplicació haurà de recollir dintre d'una base de dades totes les avaries, peticions, actualitzacions i modificacions del sistema informàtic del client. Amb aquesta base de dades s'ha de poder consultar, modificar i esborrar les dades de qualsevol operació que tingui relació amb qualsevol equip informàtic final de la xarxa del client, a més de poder obtenir un resultat gràfic d'ordre cronològic, amb l'objectiu d'obtenir estadístiques en relació al temps. Aquest projecte es distribueix en els apartats següents:

En el segon capítol tractarem dels antecedents de l'aplicatiu. Descriurem breument un aplicatiu anterior que s'utilitzava per a la mateixa funció, les seves limitacions i l'escenari de funcionament de la nova versió. En el tercer capítol veurem les necessitats de l'aplicatiu, que son els requeriments per part del client per a desenvolupar la solució i que aquesta s'adeqüi a les seves necessitats. Aquestes premisses determinaran el desenvolupament de l'aplicació, i marcaran els objectius de la mateixa. En el quart apartat es descriuen els motius de la solució adoptada, i les seves especificacions, a més de una breu descripció de les altres solucions possibles. En el cinquè apartat s'explica el disseny de l'aplicació i el seva planificació, això inclou la convergència entre les premisses i les possibilitats de desenvolupament de la solució. En el sisè apartat s'explica el propi procés de desenvolupament de la aplicació, amb un ordenament cronològic de la seva evolució, amb una recopilació dels problemes més destacats en termes de desenvolupament. El setè capítol descriu el procés de comprovació de la aplicació i el serveis que ofereix, el procés que s'ha seguit i els resultats obtinguts. El vuitè capítol correspon a les conclusions, on es recullen les observacions i opinions extretes del procés de desenvolupament. Aquí valoren tant el procés de desenvolupament com la planificació del mateix. Finalment, l'annex completa la descripció del treball realitzat i la informació emprada: Manual d'usuari, Manual del programador, Bateria de proves realitzades, etc.

Aquest informe que descriu el procés de desenvolupament d'una aplicació com a substituta d'una altra, representa el TFC de l'estudiant.

# 2 ANTECEDENTS

## *2.1 Descripció de l'aplicatiu anterior*

L'aplicació anterior estava basada en Visual Basic, i gestionava les dades contra un servidor de base de dades Access, propi de Microsoft. Aquesta aplicació tenia la funció de gestionar les incidències de la xarxa informàtica anterior. S'utilitzava en un PC i enregistrava tota la activitat del servei de Helpdesk. A la vegada, s'incorporava un web dinàmica sobre IIS i ODBC, que permetia als usuaris de la xarxa comunicar les seves incidències i que fossin enregistrades en la base de dades.

En el diagrama 1 es resumeix el procés de evolució de les incidències en l'aplicatiu anterior des de la comunicació de la incidència, fins a la delegació de la mateixa al departament corresponent.

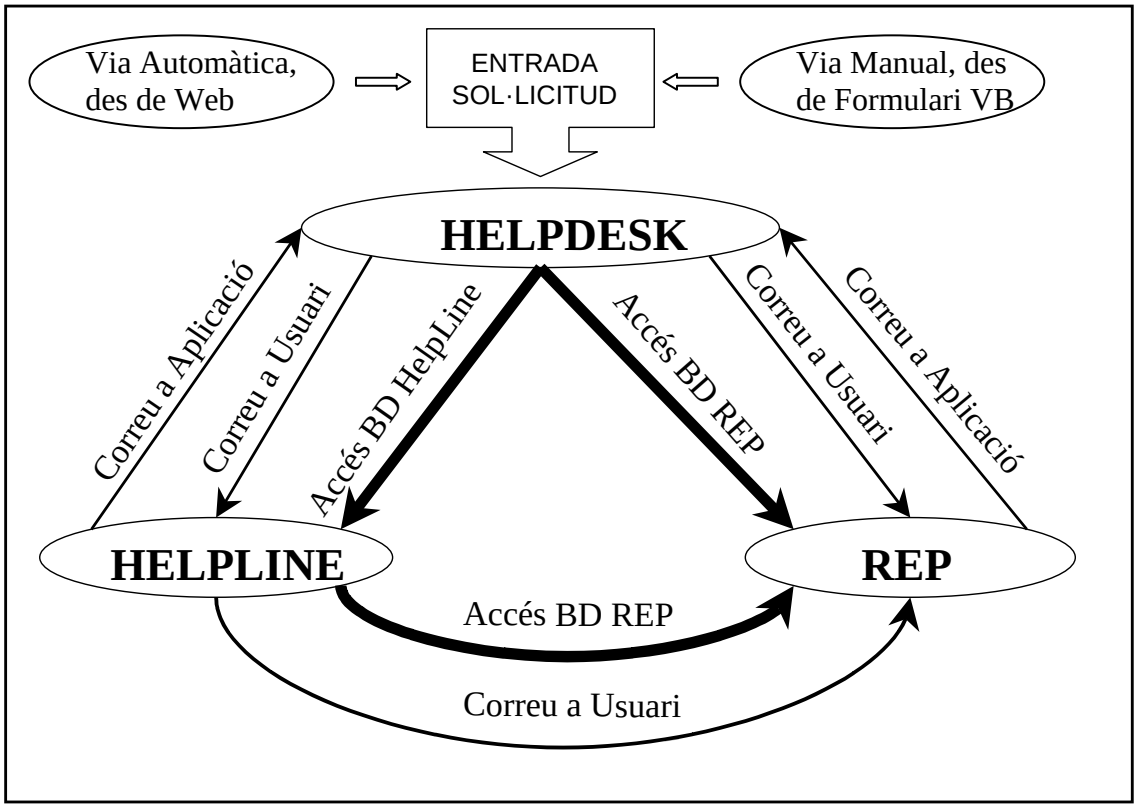



**Diagrama 1. Representació d'estats de l'aplicatiu anterior.**

El punt de partida de l'aplicatiu es troba en la introducció d'incidències. Aquestes es poden tramitar per dues vies. La primera és des de l'aplicatiu central, mitjançant la comunicació al servei Helpdesk, aquest utilitzant el formulari VB, i transmetent la incidència a la persona d'atenció a l'usuari. L'altra via, es utilitzant la web d'accés a Helpdesk, que també permet la introducció de les incidències. Aquestes sol·licituds es guarden i son ateses pel personal de Helpdesk. Un cop identificat i estudiat el problema, es deriva al departament d'ajuda en línia (Helpline) o bé al departament de reparacions (Rep), en funció de l'atenció requerida. A partir d'aquesta derivació es comunica a l'usuari l'acció presa, i s'informa al departament de les dades de la incidència. Es dona el cas que incidències ateses per Helpline, que en un principi tenen un caràcter informatiu, poden ser promocionades cap al departament tècnic de reparacions, sense haver de tornar al servei de Helpdesk. En aquesta mecànica, l'aplicatiu realitza les tasques de emmagatzematge, i comunicació entre els departaments. Per tant, l'aplicatiu desplega una interfície per a que els operaris de Helpdesk introdueixen les dades de la incidència i del usuari afectat. Aquestes dades entren a la base de dades, la qual es accessible

en tot moment pels departament. Mitjançant un correu electrònic, s'informa al departament que hi ha una nova incidència a la base de dades, i es proporciona les dades necessàries per emprendre la intervenció. Un cop aquesta s'ha realitzat, s'informa a Helpdesk mitjançant un correu electrònic de tornada.

L'esquema de la lògica de formularis Visual Basic de la aplicació anterior és la següent, que es reflexa en el Diagrama 2:

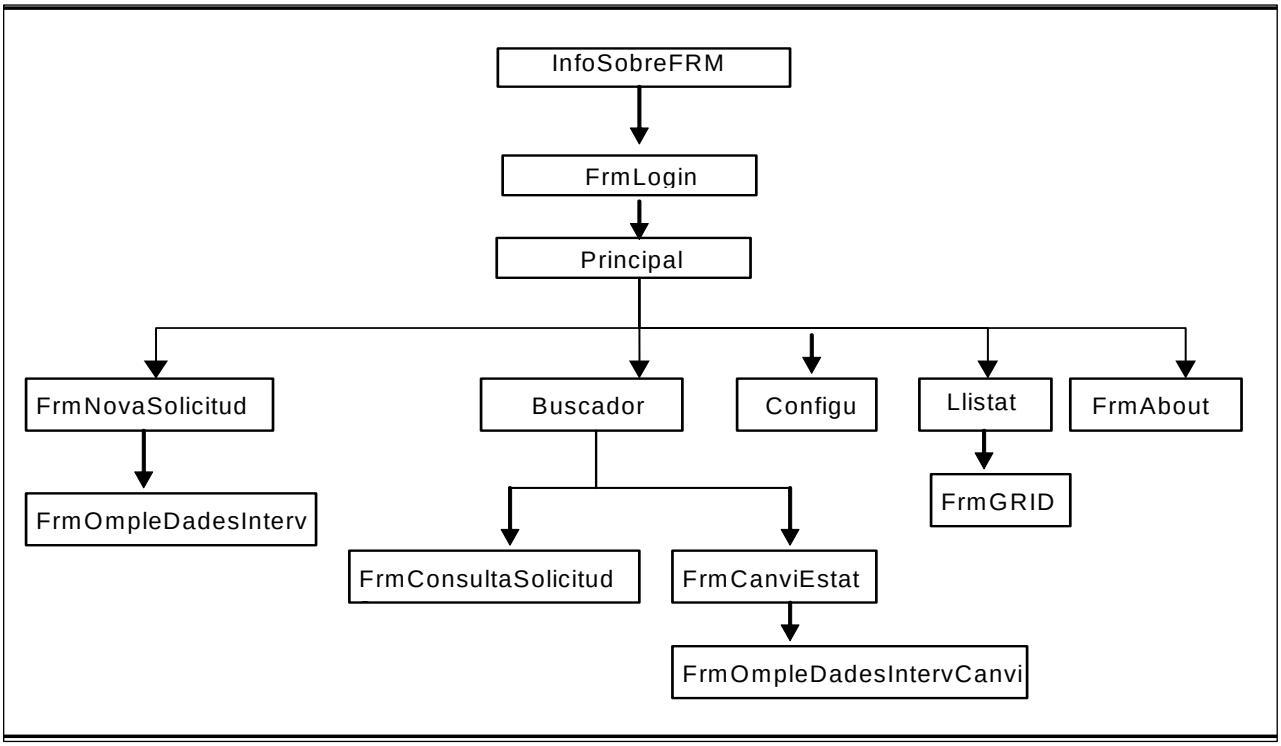
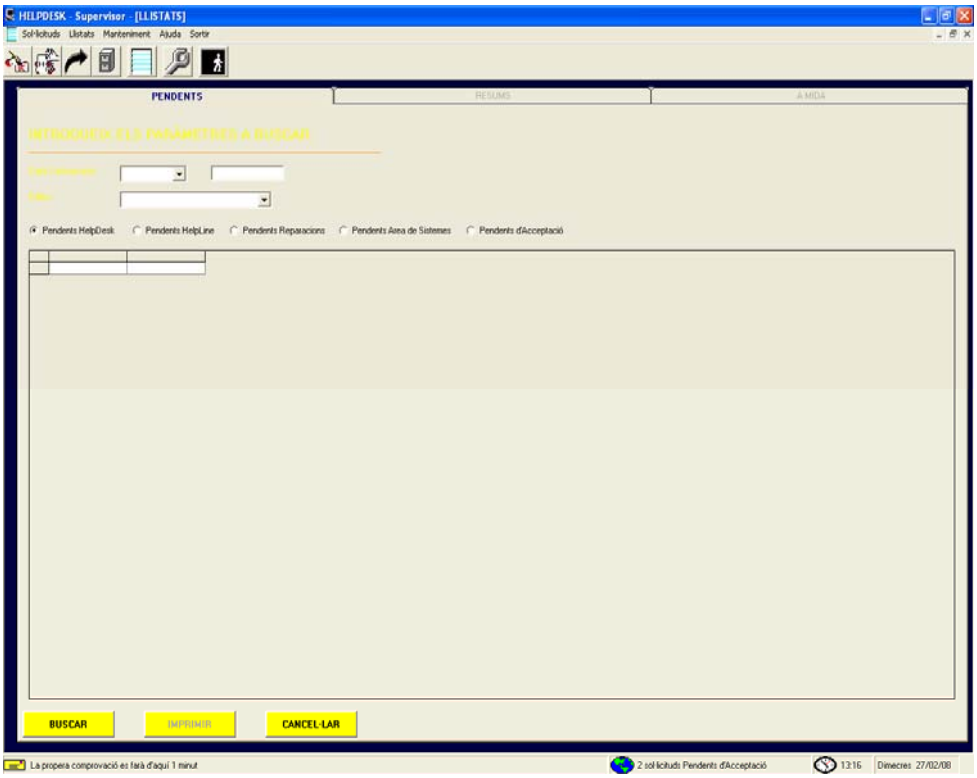



**Diagrama 2. Representació jeràrquica de formularis de l'aplicatiu anterior.**

Podem veure el funcionament bàsic de l'aplicatiu, basat en un formulari d'autenticació, des del qual es mostra el formulari principal, i des d'aquí podem accedir als formularis de nova sol·licitud, de cerca de consultes, de configuració de l'aplicatiu, de llistat de consultes i el d'informació de l'aplicatiu (About). Més en detall, la descripció de la lògica és la següent: després del formulari de Nova Sol·licitud, trobem el formulari de OmpleDadesInterv, que permet la introducció de les dades de les Intervencions fetes en funció de la sol·licitud. Per altra banda, després del formulari de Buscador, podem accedir o bé al formulari de ConsultaSol·licitud o bé al de CanviEstat, des del qual, si s'escau, anirem al formulari de OmpleDadesIntervenciónCanvi, si al canviar l'estat de la sol·licitud, hem d'alterar les dades de la intervenció que s'associa a la mateixa.

La següent captura mostra el formulari de cerca de consultes pendents de solució.




**Captura 1. Formulari de cerca**

L'aplicatiu també desplega una interfície web que permet als usuaris accedir remotament a l'aplicatiu, i accedir a la base de dades a través de la web. Aquesta funcionalitat es va preparar per als tècnics, per que poguessin accedir a les dades des de qualsevol punt de l'organització. La interfície web implementa un accés senzill a la base de dades, on destaquen la consulta, modificació i addició de dades. Mitjançant aquestes dues interfícies, s'implementava el servei de Helpdesk, que era gestionat pel conjunt dels departaments. La web del servei Helpdesk segueix l'esquema que es mostra en el diagrama 3:

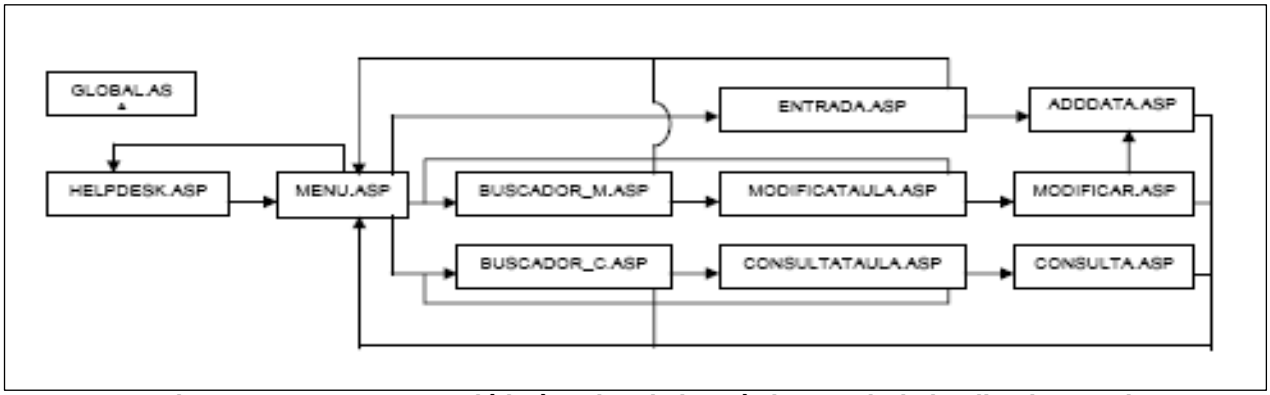



**Diagrama 3. Representació jeràrquica de les pàgines web de l'aplicatiu anterior.**

Podem veure com la pàgina inicial és Helpdesk.asp, des de la qual si ens autentiquem entrem al menú principal, Menu.asp. A partir d'aquí podem entrar noves sol·licituds i buscar consultes, bé per llistar la informació o bé per modificar les dades de la mateixa. Tant si optem per introduir noves dades com modificar les existents, el sistema ens conduirà a la pàgina Adddata.asp, que és un formulari on es recullen les dades. Per altra banda, si consultem una sol·licitud, l'aplicatiu ens condueix a una pàgina de representació de les dades anomenada Consulta.asp

## *2.2 Motivació i objectius del nou aplicatiu*

El departament d'informàtica conté dintre seu un servei de Helpdesk amb el que atendre les incidències que puguin ocórrer en la xarxa, amb la intenció de tenir màxima disponibilitat dels seus recursos informàtics. Aquestes incidències poden ser, des de consultes sobre el funcionament d'un programa, fins a la sol·licitud de dades sobre la configuració de la xarxa o bé la petició de equipament nou. Els usuaris només han de posar-se en contacte amb el servei per demanar assistència tècnica. Amb aquest servei, el departament d'informàtica recull les incidències que es produeixen, les classifica, demana les actuacions pertinents i duu un seguiment de la solució de les mateixes i de les pròpies incidències. Per facilitar les tasques del personal de Helpdesk, es va dissenyar un aplicatiu, amb el que portar un registre de les incidències i comunicar-les als tècnics.

Durant els últims temps s'ha dut a terme un canvi en la xarxa del client. Actualment, la xarxa està dividida en subxarxes, s'han establert serveis per als usuaris, i s'ha centralitzat la gestió de dades. El servei Helpdesk vol millorar la seva resposta per afrontar les incidències, i alhora, gestionar un equip de treball encarregat de desplaçar-se al lloc de la incidència i aplicar la solució. Aquest projecte té la motivació en la substitució d'un aplicatiu anterior que s'utilitzava en el departament d'informàtica, i que estava dedicat a la gestió d'incidències de la xarxa.

El client desitja incorporar millores al software de gestió de les incidències. Dissenyar un nou aplicatiu que haurà d'incorporar la generació automàtica d'informes que s'enviaran a través del correu electrònic, haurà d'incorporar una secció de representacions gràfiques de les activitats del servei de helpdesk i a més, haurà d'incorporar la característica de ser multiusuari.

Durant l'explotació s'han detectat possibles millores i detalls que s'han de ser corregits. La detecció d'aquests detalls esdevé una de les motivacions del nou aplicatiu. Una de les coses que s'havien de corregir era l'origen de dades de l'aplicatiu. El sistema engloba els departament de Helpdesk, Helpline i Servei Tècnic. Això comportava que si un departament modificava les dades de l'altre departament, provocant confusions i pèrdues d'informació, que havien de ser resoltes a través d'altres canals de comunicació. Per tant es va vol separar Helpdesk de la resta de departaments tècnics, alhora que aquest es va fusionar amb Helpline, de manera que són les mateixes operadores de Helpdesk que donen resposta a consultes o incidències senzilles. Amb aquesta separació de dades permet que la base de dades de sol·licituds passi a estar a mans únicament a Helpdesk, i el Servei Tècnic disposi de la seva pròpia base de dades, el que li permetrà un servei més eficient, gràcies a que es centra en les dades tècniques de l'equipament. A més, l'agrupament i simplificació del procés de solució des del punt de vista de l'usuari en funció de la seva complexitat, possibilita que l'usuari obtingui la resposta més efectiva.

Alhora s'aprofita per afegir funcionalitats a l'aplicatiu, en el sentit que es va voler que des de l'aplicatiu es poguessin generar una sèrie d'informes i llistats amb l'objectiu de facilitar la interpretació de les dades referents a la gestió de les incidències.

Un altre dels motius és el canvi de plataforma interna del client. L'aplicació anterior, com ja hem dit, penjava d'un servidor amb un servei de ODBC per engegar la interfície web. D'acord amb les noves directives tècniques, aquesta aplicació tenia que ser migrada al nou servidor de base de dades Oracle. Per altra banda, amb la intenció que l'aplicatiu el poguessin fer servir diverses persones a la vegada es va decidir per utilitzar un servidor d'aplicacions Web, amb capacitat de gestió de bases de dades.

# 3 REQUERIMENTS DEL NOU APLICATIU

A l'hora de dissenyar l'aplicatiu el client ha transmès els requeriments mitjançant una carta amb les especificacions funcionals que ha de tenir. Les incidències reportades pels usuaris en relació als sistemes d'informació es gestionen mitjançant una aplicació informàtica (Helpdesk) que requereix unes noves funcionalitats. L'aplicació ha de permetre gestionar de manera diferenciada les consultes, les incidències i les peticions que arriben al Departament pels diferents canals establerts a aquests efectes (correu electrònic, trucades, ...)

**Consultes:** Registrar les consultes que els usuaris realitzen al servei de Helpdesk i que es solucionen telefònicament. Les dades necessàries han de ser les mínimes, nom i cognoms de la persona que fa la consulta, la seva unitat orgànica, la tipologia de la consulta i solució. Una vegada resolta, es tanca o dóna lloc a una incidència o petició. Si és així, s'ha de permetre mantenir les dades ja introduïdes, excepte en el cas que sigui una petició que requereixi la comunicació per part del cap de la unitat corresponent.

**Incidències:** S'ha de diferenciar entre incidències de primer nivell que són resoltes directament pel servei de Helpdesk i incidències de segons nivell que són redirigides a d'altres àrees.

Incidències de primer nivell: es registra la persona que comunica la incidència (nom i cognoms, unitat orgànica, telèfon, correu electrònic), la descripció de la incidència i la solució aplicada pel servei de Helpdesk. El fet de registrar la incidència ha de generar un correu electrònic que s'enviarà a la persona que l'ha obert on se li comunica el numero d'incidència assignat i que servirà per a consultes posteriors. El fet de solucionar una incidència suposa un vist i plau de l'usuari que permetrà fer el tancament definitiu.

Incidències de segon nivell: són les que no poden ser resoltes directament pel servei de Helpdesk i són redirigides a les diferents àrees del Departament: Sistemes, Manteniment o Programari.

El procediment per a la gestió d'aquesta tipologia d'incidències és el mateix que en el cas anterior però permetrà assignar-la a una persona de les àrees esmentades i s'enviarà a través del correu electrònic amb els documents annexos que calgui.

**Peticions:** L'única diferència entre aquest circuit i l'anterior és que les persones autoritzades per tramitar peticions són els caps de les unitats corresponents.

Tot això fa que una necessitat important en la gestió de les incidències sigui el mòdul de cerques que ha de permetre fer-les en funció de:

- Numero d'incidència/petició
- Data d'incidència
- Persona Responsable
- Àrea
- Estat

Així mateix, l'aplicació ha de permetre l'elaboració d'informes estadístics (taules i gràfics) de la tipologia següent:

Informes mensual indicant el total de consultes, incidències i peticions.
Informes mensuals de les incidències diferenciant entre les de 1r i 2n nivell.
Informe acumulatiu anual.

Per altra banda, el client va especificar la plataforma sobre la qual s'haurà de desenvolupar l'aplicatiu. Aquest tenia que desenvolupar-se sobre l'arquitectura de aplicació Web, basat en un servidor d'aplicacions Java, concretament el BEA Weblogic 9.2. Per gestionar les dades introduïdes, s'utilitzarà el gestor de bases de dades Sun Oracle 8.1i. Tots dos escollits per la seva capacitat de conviure amb altres serveis del client, treballant de forma paral·lela però compartint recursos.

De manera paral·lela, el següent diagrama mostra clarament la classificació dels estats en funció del seu procés d'evolució, que s'agrupa en Pendent de solució, Solucionat, No Solucionat i Conformitat. El primer estat representa l'estat previ a l'actuació, el segon estat representa l'aplicació de una solució, i la validació per part dels tècnics. El tercer estat representa l'aplicació de la solució, però que no comporta la validació per part del tècnic, sense tenir la conformitat del usuari. Tant per motius de manca de comunicació, o bé per la no conformitat i finalment, el quart estat és l'estat que representa la validació i  a la vegada, la conformitat del usuari. Aquesta descripció queda resumida en el diagrama número 4.

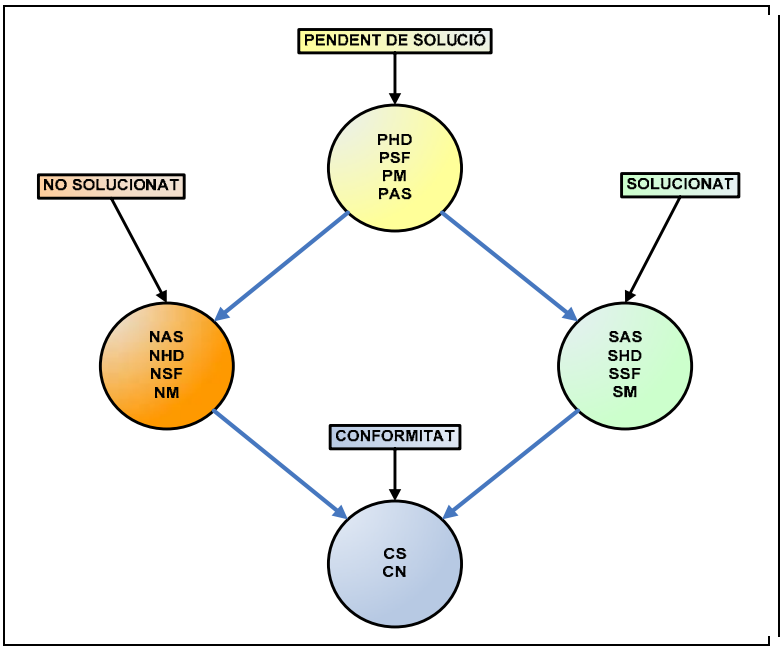



**Diagrama 4. Representació dels estats de la nova aplicació**

# 4 DISSENY DE LA SOLUCIÓ

## *4.1 Plantejament inicial del disseny*

1. Definició dels objectius del client, en concret dels usuaris finals. És molt important saber quin enfocament es vol donar a l'aplicatiu, quins són els detalls rellevants. Entre aquests podem destacar la velocitat, la facilitat d'ús, etc. i quin termini es disposa per desenvolupar l'aplicatiu permet dimensionar la feina amb major claredat.

2. Coneixement de l'entorn de desenvolupament i les eines relacionades amb el propi desenvolupament.

3. Definició de les parts en que es divideix el software, des del punt de vista de la programació. És a dir, en disseny de l'aplicació, l'estructuració prèvia arran del software de desenvolupament condiciona en gran part el disseny d'aquesta, i de la mateixa manera, concretar l'estructura immediatament després del coneixement de la dinàmica de desenvolupament posa les bases per a la producció posterior.

4. Desenvolupament de les classes definides. Un cop definides les bases del software, a partir de l'entorn de programació i del client, es passa a programar, en temps de disseny, l'entorn de programació, definint proves de funcionalitat continues.

5. Comprovació final de les funcionalitats de l'aplicatiu, la creació de la documentació i la formació dels futurs usuaris d'acord amb el client, a més de la instal·lació i la posada en funcionament de l'aplicatiu.

## *4.2 Condicions restrictives*

Per a la definició de l'estructura s'han de tenir en compte les premisses següents :

- Dissenyar una **estructura que optimitzés recursos**. L'entorn BEA Weblogic és un entorn corporatiu, que permet els serveis distribuïts en diversos servidor, balanceig de carrega, capes d'adaptació entre diversos serveis, etc. Per diferents proves s'ha comprovat que la optimització és crucial, i per tant, s'ha dedicat una major atenció al codi.

- Dissenyar objectes lògics, funcionals, i evitar les classes intermèdies. Construir classes que representessin consultes, e-mails, llistats, però evitar l'ús en la mesura del possible de classes com ara Opcions de correu.

- Establir una connexió adient amb la **base de dades**. L'entorn BEA Weblogic proposa una connexió JDBC a través de noms Java (JNDI), configurats dins del que s'anomena dominis. Per a la execució de codi SQL contra aquesta connexió s'havien d'utilitzar classes amb uns mètodes preestablerts.

- La **visió de web**, el que condiciona alhora d'estructura els procediments, i els formularis d'informació. Les accions a realitzar es tradueixen en transicions entre pàgines JSP.

- La **accessibilitat**. A cada secció (Consultes, Incidències, etc.) les funcions de cerca i gestió tenen que ser accessibles.

Finalment, un cop enumerats els mètodes, les classes, les pàgines que serien necessàries, en un primer moment, per implementar l'aplicatiu, es va passar a fer proves amb l'entorn de desenvolupament, per tal de poder implementar aquests mètodes i classes. S'ha de tenir amb compte que al mateix temps que s'ha anat provant les possibilitats de l'entorn, s'han anat modificant les implementacions, en busca d'oferir la mínima quantitat de instruccions necessàries per realitzar un mètode o acció.

## *4.3 Valoració econòmica*

Per a la estimació dels costos de desenvolupament, classificarem les fases de desenvolupament en les tasques que s'han de realitzar:

- Especificació de l'aplicatiu: En aquest apartat s'engloben les tasques d'identificació de les necessitats del clients, l'obtenció i configuració de les eines de desenvolupament.
  Estimació d'hores: 32 hores.

- Disseny de les classes, taules i altres entitats: En aquest apartats es duen a terme el disseny de la base de dades de l'aplicatiu i de les classes que han de relacionar la informació real dintre l'aplicatiu. Estimació d'hores: 16 Hores.

- Disseny dels mètodes: En aquest apartat es classifiquen totes les tasques de desenvolupament de funcions, mètodes i altres automatitzacions necessàries per a l'execució de l'aplicatiu.
  Estimació d'hores: 80 Hores.

- Implementació de l'aplicatiu: En aquest apartat estan les tasques finals de programació, en que es poleix el codi en busca d'errades de codificació que afectin per exemple a formatació de dades, etc. Estimació d'hores: 160 Hores.

- Comprovació de resultats: Aquí es computen els temps de compilació, d'execució de proves, etc. Estimació de hores: 40 Hores.

- Entrevistes per a la definició de requeriments.
  Estimació de hores: 16 hores.

- Realització de la documentació associada.
  Estimació de hores: 80 hores.

- Formació dels usuaris.
  Estimació de hores: 4 hores.

- **Estimació total de hores: 428 Hores * 45€ / Hora = 3936 €**

# 5 PROCÉS DE DESENVOLUPAMENTS DE L'APLICATIU

## *5.1 Accions prèvies*

La primera tasca, com ja hem dit, és definir amb el client el projecte i els seus objectius. Mitjançant una entrevista s'ha d'intentar reunir el major nombre de detalls al respecte. Sense aquesta informació es molt fàcil perdre temps implementant classes, mètodes o pàgines inútilment, ja que el client pot desestimar-les sense apreciar la possible funció que tinguin, pel simple motiu de que els seus objectius siguin uns altres. Com hem dit, mitjançant una entrevista podem recollir la majoria de la informació necessària. El més important és passar per una sèrie de conceptes i aconseguir que el client opini sobre els punts: nombre de funcions que necessita, forma de presentació de dades, nombre de usuaris, situació lògica de l'aplicatiu dintre la xarxa, entorn de funcionament, disponibilitat i seguretat són alguns dels conceptes més importants que s'han d'intentar aclarir en una entrevista amb el client.

En la majoria dels casos els clients tenen preparats els requeriments del programa, tot i que en alguns casos és necessari preguntar per algun punt, de manera que es pugui començar a dissenyar el programa en la direcció correcta.

La segona tasca per utilitzar l'entorn de desenvolupament es segueix els tutorials que s'adjunten en la versió de desenvolupament del servidor d'aplicacions BEA Weblogic. El que es segueix es una aplicació Web amb connexió amb una base de dades. Aquest tutorial mostra com connectar el projecte amb la base de dades, des del workspace fins al servidor d'aplicacions configurat per a aquest workspace. El següent pas es anar afegint el que s'anomena "accions". Un cop es tenen unes bases per desenvolupar les pròpies accions, pàgines, controls, etc, el següent pas es configurar el workspace i el server d'acord amb la configuració de la base de dades.

## *5.2 Model d'aplicatiu Web sobre BEA Weblogic 9.2*

Aquest servidor de pàgines funcionament de la plataforma sobre la qual s'ha de desenvolupar l'aplicatiu es basa en quatre parts indispensables: el controlador de flux, el control, les dades (driver de la base de dades) i pàgines. Partint de la interfície amb l'usuari, que és la pàgina HTML, la interacció amb el aplicatiu segueix el següent patró:

- L'usuari selecciona un link, fa clic a un botó o valida un formulari.
- Es genera una petició HTTP, amb la informació del link/botó/formulari dintre les capçaleres.
- El servidor Web, que en aquest cas fa de missatger, rep la petició, en separa les capçaleres i envia les dades al controlador de flux.
- El controlador de flux rep les dades de les capçaleres i les relaciona com a mètode que ell mateix conté a cridar i paràmetres del mateix.
- El mètode del controlador de flux crida a un control, que implementa mitjançant una estructura la connexió a la base de dades, la consulta que se li ha de fer i la gestió de les dades de retorn.
- El control li retorna les dades i el control al controlador de flux.

- El controlador de flux, amb les dades tractades, crida a una pàgina amb la qual les representarà, presentant un formulari amb el qual reiniciar el procés.

En el diagrama 7 podem veure gràficament la representació del flux de control de l'aplicatiu entre els diversos components que el formen, durant l'execució d'una acció des del moment en que fem activem un objecte a la pàgina JSP, fins que visualitzem al navegador la resposta en forma de pàgina JSP nova.

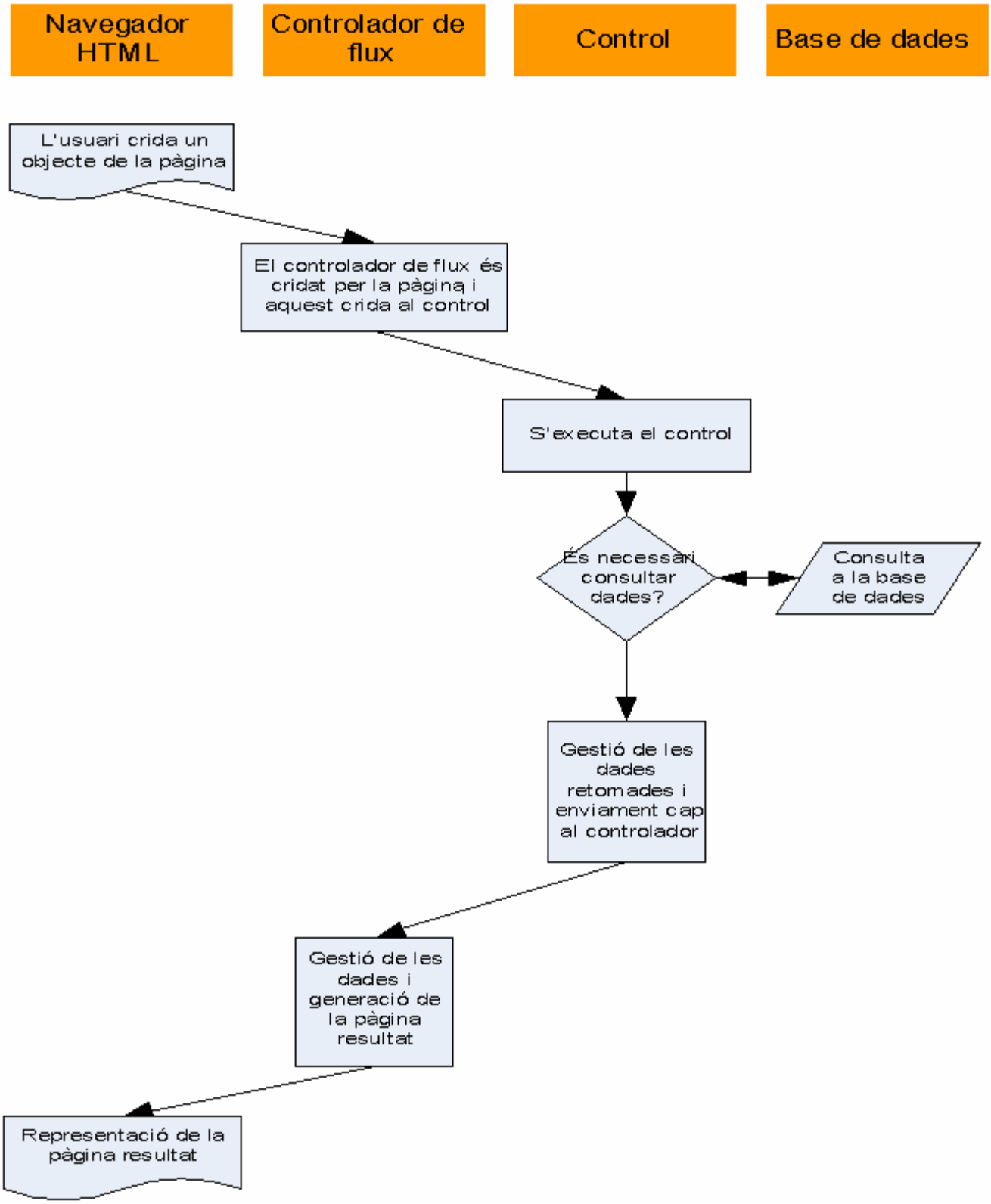



**Diagrama 5. Lògica de funcionament de l'aplicatiu Web.**

**Controlador de flux**: La lògica del controlador de flux es basa en accions(mètodes) que són implementades en el controlador de flux, i cridades per les pàgines. Per interaccionar amb la base de dades recorre a les implementacions de controls. D'aquesta manera, el controlador de flux es mostra com el nexe d'unió de les diferents lògiques que implementaran l'aplicatiu. El seu paper principal és

que les pàgines tinguin la menor quantitat possible de codi, de manera que aquest codi sigui el més versàtil possible.

Les accions són mètodes i es declaren com a tals, però aquestes tenen el propi nom d'accions. Quan es crida a una acció, aquesta té com a entrada els paràmetres rebuts en la petició HTTP, per relacionar-lo en variables s'utilitza una classe específica anomenada FormBean. Aquesta classe FormBean es basa en la creació una propietat que es correspon amb la classe que s'espera dintre l'acció, i té com a mètodes set i get, que són utilitzats per Beehive per donar forma als paràmetres de la petició HTTP dintre una classe que pugui ser tractada per l'acció i a l'inversa, introdueix en el codi HTML encapsulat dintre el paquet HTTP els valors de les classes. Aquesta, per tal de completar la transició, al final del codi s'ha de retornar un objecte anomenat Forward, que conté la URL de destí i, de manera opcional, un conjunt de ActionOutputs, que tenen molta similitud amb els paràmetres típics que es retornen en un mètodes o resposta HTTP. Les URL poden ser accions dintre del propi controlador, o bé serveis externs que escolten la capa HTTP, com per exemple podria ser un Servlet. Aquests ActionOutputs poden tenir la forma que es desitgi, classes primitives, classes d'usuari, etc.

Per implementar les accions, primer que res s'ha d'incloure els "imports" necessaris, en el nostre cas, apart de cridar a les llibreries intrínseques de l'entorn, com pot ser la llibreria org.apache.beehive.*, s'ha de cridar als models de dades, per exemple, Consulta, Incidència, Correu, etc. Un cop realitzada aquesta crida, la definició de la pròpia classe del controlador de flux, podrem crear les accions, que definiran les transicions entre pàgines JSP. Dintre de l'acció del controlador de flux, es pot observar que es realitzen les tasques de autenticació i gestió d'errors. A més, es crida al codi corresponent al FormBean que relaciona el formulari i que es passa com a paràmetre a l'acció:

```
@Jpf.Action(forwards={
        @Jpf.Forward(name="page1", page="page1.jsp", actionOutputs={
                @Jpf.ActionOutput(name = "Bool", type = "java.lang.Boolean.class")},  @Jpf.Forward(name="page2",
        page="page2.jsp", actionOutputs= {
                @Jpf.ActionOutput(name = "Bool", type = "java.lang.Boolean.class")  }}))
public Forward someAction(FormBean myBean)
{
        if (control.checkUser(myBean.getUsername(),myBean.getPassword())
        {
                Forward F = new Forward("page1");
                F.addActionOutput("Bool",true);
                return F;
        }
        else
        {
                Forward F = new Forward("page2);
                F.addActionOutput("Bool",false
                return F;
        }
}

@Jpf.FormBean

public static class myBean implements
 java.io.Serializable
{

        private static final long serialVersionUID = -191195858L;

        public MyFormBean() {
                String username,
                String password;
        }

        public bool getUsername() {
                return username;
        }
```

```
		public void setUsername(String u) {
			this.username = u;
		}

		public bool getPassword() {
			return passsword;
		}

		public void setPassword(String p) {
			this.password = p;
		}
	}
```

**Implementació de les funcionalitats mitjançant les classes Controls:** Per al modelat de les dades, i la implementació dels mètodes, s'utilitza un conjunt de classes que proporcionen els mètodes necessàries, i que inclouen les sentencies SQL contra la base de dades.

**Control.java**: En aquest fitxer es defineixen els mètodes, els tipus i els paràmetres. Aquestes definicions també són usades per BEA Workshop per mostrar els mètodes utilitzables en les accions. Per exemple, un mètode dintre aquesta classe es codifica de la següent manera :

```
package controls;
import org.apache.beehive.controls.api.bean.*;

@ControlInterface
public interface Control
{
	public bool checkUser (String username, String password);
}
```

**ControlImplementation.java** : Conté les implementacions de cadascun dels mètodes llistats en la classe Control: Es tracten els paràmetres, i en aquesta classe és on s'interacciona amb la base de dades a través de la crida de mètodes que executen les sentències SQL. En l'ús d'aquests mètodes és imperatiu contenir-lo dintre de blocs del tipus "try-catch".Seguint amb l'exemple anterior :

```
package controls;
import org.apache.beehive.controls.api.bean.*;
import java.io.Serializable;

@ControlImplementation
public class ControlImpl implements Control, Serializable
{
	@Control()
	Private DB db;

	public bool checkUser(String username, String password)
	{
		try
		{
			return db.checkUser(username, password);
		}
		catch(Exception e)
		{
			System.out.println(e.toString());
			return null;
		}
	}
}
```

Aquest codi implementa un senzill mètode, que retorna un booleà retornat pel mètode checkUser en funció de si existeix l'usuari, que per la codificació es crida contra la base de dades. Per a accedir a tal es defineix un control que connecta amb el següent fitxer que implementa l'accés a la base de dades.

**DB.java** : Aquest fitxer conté mètodes específics per a utilitzar sentencies SQL, aquests no són accessibles des de cap altra classe excepte en la Implementació de Controls. Aquests mètodes es basen en la definició de sentències i la definició del mètode que el crida. Això és possible a través de l'ús de la capa de noms JNDI, que crida a la base de dades a través d'un driver extern i transparent al programador, d'aquesta manera s'estalvia temps en codificar la connexió, que en quasi la majoria d'aplicacions desenvolupades, és el mateix codi. Com a part final de l'exemple anterior, seguidament definirem el mètode checkUser.

```
package controls;
import org.apache.beehive.controls.api.bean.*;
import org.apache.beehive.controls.system.jdbc.*;

@ControlExtension
@JdbcControl.ConnectionDataSource(jndiName = "driver1")
public interface DB extends JdbcControl
{
        @JdbcControl.SQLstatement("EXISTS * FROM USUARI WHERE NAME= {username} AND PASSWORD = {password}");
        public bool checkUser(String username, String password);
}
```

Aquest codi implementa la consulta a través del llenguatge SQL: es defineix el driver de connexió, i la sentència SQL, amb la sentencia EXISTS únicament per confirmar l'existència d'un registre en que concordi el nom i el password facilitats.

**Pàgines JSP i la seva relació amb el Controlador de Flux.** En el desenvolupament de les pàgines JSP, s'utilitza codi Java en format d'etiquetes HTML. L'ús de Apache Beehive i Apache Struts afegeix una sèrie de etiquetes que proporcionen una connexió de entrada i sortida de dades entre les pàgines JSP i el controlador de flux. A més, es permet l'ús de les classes implementades i que s'utilitzen en la resta de l'aplicatiu. Després d'haver construït les classes, l'entorn de desenvolupament permet crear pàgines JSP destinades a interaccionar amb el controlador de flux, codificant automàticament datagrids, formularis i altres formes d'entrada/sortida de dades. Així, es possible desenvolupar ràpidament formularis que permeten introduir dades al aplicatiu, i generar pàgines JSP amb llistats o datagrids amb els quals visualitzar les dades de sortida.

Les accions del controlador de flux s'apunten a una pàgina JSP, per poder mostrar en la finestra del navegador els formularis destinats a l'entrada de dades a l'aplicatiu, i els resultats o sortida de les mateixes un cop han estat tractades per la lògica de l'aplicatiu.

Per poder enviar contra les pàgines JSP les classes que volem representar, s'utilitza una definició anomenada ActionOutputs, on s'especifiquen el nom de la classes i el tipus de la classe, que per exemple pot ser String, String[], o bé, models.Consulta. Per a enviar des del controlador de flux a la pàgina JSP, s'ha d'incloure la següent sentència en la declaració de l'acció.

```
@Jpf.ActionOutput(name = "Mostra", type = "String.class")
```

Tanmateix, en la pàgina JSP i mitjançant l'entorn hem de declarar el que s'anomena PageInput, mitjançant la etiqueta següent, per poder-la representar en la pàgina, dintre de qualsevol de les possibilitat que se'ns ofereix:

```
<netui:declarePageInput name="Mostra", type="String" />
```

A partir d'aquest moment, podem utilitzar la variable Mostra bé com a valor dintre de formularis com a valor per defecte, o bé per mostrar com a text incorporat dintre la pàgina JSP. Pel cas de classes complexes, com podria ser Consulta, o bé Incidència, BEA Workshop proporciona un constructor de datagrids i datalists, que ens permeten crear taules i llistats per mostrar les dades.

El primer ens permet llistar les propietats de una classe o bé d'una matriu de classe, a més de generar columnes que contindran vincles que permetin cridar a accions, possibilitant passant un o

més paràmetres. D'aquesta manera podem fer datagrids que per aquests cas ens permeten llistar consultes i incidències amb enllaços per portar les consultes als formularis de modificació o bé per eliminar-les, per exemple.

Els datalists són molt similars, però no incorporen aquesta funcionalitat de generar vincles. Aquests llistats s'implementen automàticament amb etiquetes pròpies de Apache Beehive, i tenen un comportament molt similar a les etiquetes HTML de llistats com ara <TR> o <TH>.

```
<netui-data:dataGrid name="datagrid1" dataSource="pageInput.dades">
        <netui-data:configurePager disableDefaultPager="true" />
        <netui-data:header>
            <netui-data:headerCell headerText="camp1" />
        </netui-data:header>
        <netui-data:rows>
            <netui-data:spanCell value="${container.item.camp1}">
            </netui-data:spanCell>
        </netui-data:rows>
</netui-data:dataGrid>
```

Quan s'obri la pàgina JSP dintre el navegador, internament el servidor llegeix les etiquetes JSP, les interpreta i injecta codi HTML que generen el resultat desitjat. A més, s'afegeix un identificador a cada camp per garantir que quan si hi ha un retorn en forma de formulari o enllaç, el controlador podrà relacionar les dades que siguin retornades.

Per entrar dades al sistema, s'utilitza un formulari JSP en la pàgina, en forma de formulari. El següent exemple mostra un formulari d'autenticació, per al mètode anterior checkUser.

```
<%@ page language="java" contentType="text/html;charset=UTF-8"%>
<%@taglib uri="http://beehive.apache.org/netui/tags-html-1.0" prefix="netui"%>
<%@taglib uri="http://beehive.apache.org/netui/tags-databinding-1.0" prefix="netui-data"%>
<%@taglib uri="http://beehive.apache.org/netui/tags-template-1.0" prefix="netui-template"%>
<netui:html>
    <head>
        <netui:base/>
    </head>
    <netui:body>
    <netui:form action="checkUser">
        <div>
            <table>
                <tr valign="top">
                    <td class="classCSS">
                        <label for="field0"> Nom: </label>
                    </td>
                    <td class="classCSS">
                        <netui:textBox dataSource="actionForm.username" TagId="field0">
                        </netui:textBox>
                    </td>
                </tr>
                <tr valign="top">
                    <td class="classCSS">
                        <label for="field1"> Password: </label>
                    </td>
                    <td class="primer">
                        <netui:textBox dataSource="actionForm.password" tagId="field1"
                        password="true"></netui:textBox>
                    </td>
                </tr>
            </table>
            <netui:button value="login" type="submit" />
        </div>
    </netui:form>
    </center>
  </netui:body>
</netui:html>
```

## *5.3 Arquitectura lògica de l'aplicatiu*

Per a la gestió de les mateixes, identifiquem les següents funcions que s'han de realitzar:

- Alta o introducció de dades dintre la base de dades
- Consulta o cerca en funció de determinats paràmetres i determinat nombre dels mateixos
- Modificació de les dades emmagatzemades
- Eliminació de les mateixes
- Gestió d'usuaris: Alta, Baixa, Modificació i Autenticació dels mateixos
- Realització d'informes de les entitats en funció de les dates parametritzades
- Generació de informes gràfics en funció dels informes anteriorment esmentats.
- Generació de correus electrònics amb un informe adjuntat.

Partint de les premisses, de les condicions restrictives partint de la base de l'estructuració de l'aplicatiu anterior, podem representar el flux de les incidències amb el diagrama següent, amb el qual es pot veure com les incidències van evolucionant cap a una solució satisfactòria o no, ja que cap la possibilitat de que no obtinguem una solució satisfactòria o no, en funció de la naturalesa de la incidència.

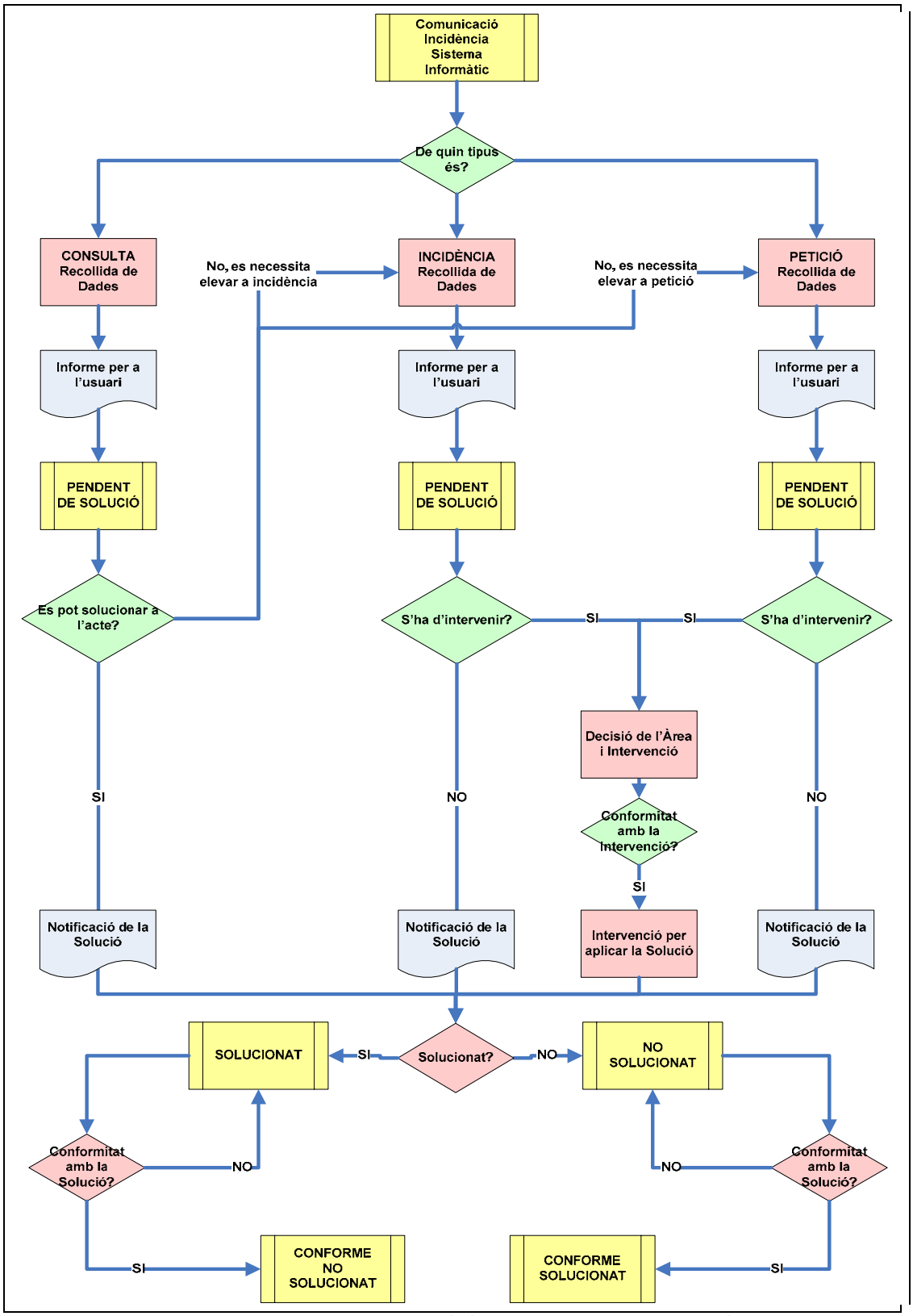



**Diagrama 6. Representació lògica de la nova aplicació**

En el diagrama 6 es mostra el procés que recorre la incidència. La fase inicial consta de l'arribada, per qualsevol dels canals possibles, al servei de Helpdesk. El mateix personal del servei, classifica la incidència en funció de la naturalesa i l'origen de la mateixa. Un cop introduïda a l'aplicatiu es genera un informe que es transmès a l'usuari afectat. Un cop emmagatzemada, passa a estar en estat

pendent de solució. El següent que s'ha de fer es buscar la solució i el mitjà de aconseguir-la, si es pot resoldre des del mateix servei de Helpdesk, o bé necessita la intervenció d'algun altre servei. Si es dóna el cas que la solució és immediata, aquesta es transmet immediatament, sinó, es transmet als departaments adequats o bé passa a ser considerada incidència. Després de l'aplicació de la solució, sigui aquesta provinent de Helpdesk, o a través del departament assignat, es considera la incidència solucionada, i el següent pas es demanar la conformitat del usuari. El final ideal es que aquesta es valori per l'usuari com a satisfactòria, però es pot donar el cas en que cap servei pugui oferir una solució que l'usuari accepti, cas que es contempla, i es considera en el procés de la incidència.

## *5.4 Classes*

Un cop analitzats els elements de l'aplicatiu, es passa a la implementació del disseny preliminar. Es divideix la descripció del desenvolupament de l'aplicatiu en parts: Classes bàsiques, classes auxiliars, base de dades, pàgines i elements externs.

### *5.4.1 Descripció de les classes i mètodes bàsics*

Per mostrar el desenvolupament de les classes i mètodes bàsics per al funcionament de l'aplicatiu, mostrarem l'exemple de l'acció d'afegir una consulta. Per mostrar com s'implementa aquesta acció, s'inclourà el codi necessari, des de la definició de la classe Consulta fins a la pàgina Web on es mostra el formulari corresponent a la introducció de Consultes. En primer lloc, llistarem la classe Consulta:

**Consulta.java**

```
package model;
import java.io.Serializable;
public class Consulta implements Serializable{

	private static final long serialVersionUID = 1L;
	private int IdConsulta;
	private String CodiConsulta;
	private String nom;
	private String cognoms;
	private String UnitatAdministrativa;
	private String Correu;
	private String Telefon;
	private java.util.Date DataComunicacio;
	private String Tipologia;
	private String Consulta;
	private String SolucioFinal;
	private String Estat;
	private java.util.Date DataConforme;
	private String UsuariResponsable;

	public Consulta(){    super();    }

	public void setIdConsulta(int IdConsulta)		{		this.IdConsulta = IdConsulta;  }

	public int getIdConsulta()	{		return this.IdConsulta;		}

	public void setCodi(String CodiConsulta)		{		this.CodiConsulta = CodiConsulta;		}

	public String getCodi()		{		return this.CodiConsulta;	}

	public void setNom(String Nom)		{		this.nom = Nom;	}

	public String getNom()		{		return this.nom;	}
```

```
	public void setCognoms(String cognoms)		{		this.cognoms = cognoms;	}

	public String getCognoms()	{		return this.cognoms;		}

	public void setCorreu(String Correu)	{		this.Correu = Correu;		}

	public String getCorreu()	{		return this.Correu;	}

	public void setTelefon(String Telefon)	{		this.Telefon = Telefon;		}

	public String getTelefon()	{		return this.Telefon;	}

	public void setDataComunicacio(java.util.Date DataComunicacio)	{
		this.DataComunicacio = DataComunicacio;
	}

	public java.util.Date getDataComunicacio()		{		return this.DataComunicacio;	}

	public void setTipologia(String Tipologia)		{		this.Tipologia = Tipologia;	}

	public String getTipologia()	{		return this.Tipologia;		}

	public void setConsulta(String Consulta) {		this.Consulta = Consulta;	}

	public String getConsulta()	{		return this.Consulta;		}

	public void setSolucioFinal(String SolucioFinal) {	this.SolucioFinal = SolucioFinal;		}

	public String getSolucioFinal() {		return this.SolucioFinal;	}

	public void setEstat(String Estat)		{		this.Estat = Estat;	}

	public String getEstat()		{		return this.Estat;	}

	public void setDataConforme(java.util.Date DataConforme)	{		this.DataConforme = DataConforme;	}

	public java.util.Date getDataConforme()	{		return this.DataConforme;	}

	public void setUsuariResponsable(String UsuariResponsable)		{
	this.UsuariResponsable = UsuariResponsable;
	}

	public String getUsuariResponsable()	{		return this.UsuariResponsable;		}

	public void setUnitatAdministrativa(String UT)	{		this.UnitatAdministrativa = UT;		}

	public String getUnitatAdministrativa()	{		return this.UnitatAdministrativa;		}

	public boolean ComprovaTancat(String[] EstatsTancats)	{
		try
		{
			for (int i=0;i<=EstatsTancats.length;i++)
			{
				if (this.Estat.equals(EstatsTancats[i]))
					return true;
			}
			return false;
		}
		catch(Exception e)
		{
			System.out.println(e.toString());
			return true;
		}
	}
}
```

Com es pot veure, la classe Consulta llista les propietats de la mateixa, i els mètodes que han de ser propis i necessaris durant el seu cicle de vida. Seguidament, mostrarem el codi que s'introdueix al controlador de flux, que com s'ha pogut veure anteriorment, és el nexe d'unió entre totes les funcionalitats de l'aplicatiu web. El seu desenvolupament es basa en l'acumulació d'accions que permetin realitzar les accions contingudes en els controls. Seguidament es mostra un exemple

d'implementació per a la introducció de noves incidències. Dintre d'aquesta acció s'utilitza un formulari en una pàgina web, on es disposen dels camps necessaris per a la validació de la incidència. Un cop les dades han estat introduïdes al formulari i es prem el botó "Afegir", el navegador envia les dades del formulari a través de la petició HTTP. En aquest cas el que es fa es introduir les dades a la base de dades, a través del mètode del control consultaControl.afegirConsulta.

**Pf2Controller.java**

```
@Jpf.Action(forwards = {
	@Jpf.Forward(name = "success", path = "ConsultaResultat.jsp", actionOutputs = {
		@Jpf.ActionOutput(name = "RConsultes", type = Consulta[].class)}),
	@Jpf.Forward(name = "error", path = "Consultes.jsp", actionOutputs = {
		@Jpf.ActionOutput(name = "Missatge", type = java.lang.String.class)
			@Jpf.ActionOutput(name="_Usuaris", type = String[].class),
			@Jpf.ActionOutput(name = "_EstatsTots",type = String[].class),
			@Jpf.ActionOutput(name = "_EstatsOberts", type = String[].class),
			@Jpf.ActionOutput(name = "UsuariActiu", type = String.class),
			@Jpf.ActionOutput(name = "_Tipologies", type = String[].class),
			@Jpf.ActionOutput(name = "_UA", type = String[].class)}),
	@Jpf.Forward(name = "exit", action = "Logout")    })
	public Forward afegirConsulta(AfegirConsultaFormBean form) {
		if (getusuari()==null)
			return new Forward("exit");
		model.Consulta consulta = form.getConsulta();
		if (consultaControl.afegirConsulta(consulta))
		{
			Forward forward = new Forward("success");
			Consulta[] proves = {new Consulta()};
			proves[0] = consulta;
			forward.addActionOutput("RConsultes",proves);
			return forward;
		}
		else
		{
			Forward backward = new Forward("error");
			backward.addActionOutput("_EstatsTots",AControl.RetornaEstats());
			backward.addActionOutput("_EstatsOberts",AControl.RetornaEstatsOberts());
			backward.addActionOutput("_Usuaris",usuariControl.retornaNoms());
			backward.addActionOutput("UsuariActiu",getUsuariActiu());
			backward.addActionOutput("_Tipologies",AControl.retornaTipologies());
			backward.addActionOutput("_UA",AControl.retornaUnitatsAdministratives());
			backward.addActionOutput("Missatge","Error amb la BBDD.");
			return backward;
		}
	}
```

Un cop es valida el formulari de la pàgina JSP, es crida a l'acció dintre del controlador de flux, que a la vegada, crida un FormBean de formulari, que li proporciona a l'acció les dades enviades amb la petició HTTP del formulari. Aquesta acció en concret realitza les següents tasques:

Comprova que l'usuari actiu tingui la sessió valida. Això es comprova utilitzant una variable del controlador de flux que s'esborra per configuració del servidor als 30 minuts. Les variables de sessió del controlador de flux, amb la configuració per defecte, es vinculen a la sessió del navegador, podent diferenciar entre dos clients Web diferents dintre la mateixa adreça IP.

Després es crida al mètode afegirConsulta, al que se li passa una Consulta, i retorna un booleà. En funció del retorn del booleà és crea un retorna de l'acció, o un altre. En cas de que el mètode afegirConsulta, èxit, es crea el Forward de retorn, i es crea una matriu de consultes buida, on s'introdueix la consulta que s'acaba de generar, a fi de mostrar-la per pantalla en la pàgina ConsultaResultat.jsp.

En cas negatiu, es retornen totes les dades necessàries per tornar a mostrar els formularis de la pàgina origen, des de la qual hem enviat el formulari amb la nova consulta. Aquesta pàgina té la

singularitat que a més incorpora un formulari de cerca de consultes. Tots dos incorporen dades que són recopilades prèviament del servidor, de manera que al formulari es mostren com a opcions. Per tant, per tornar a aquesta pàgina el que hem de fer és tornar a cridat aquesta informació i enviar-la al Forward, per què la nova pàgina tingui totes les dades necessàries per a la representació.

A continuació es mostra el codi del FormBean, que realització la transacció de dades cap a l'acció.

```
@Jpf.FormBean
	public static class AfegirConsultaFormBean implements java.io.Serializable {
		private static final long serialVersionUID = -1030312243L;
		private model.Consulta consulta;

		public AfegirConsultaFormBean() {
			consulta = new model.Consulta();
		}

		public model.Consulta getConsulta() {
			return consulta;
		}

		public void setConsulta(model.Consulta consulta) {
			this.consulta = consulta;
		}
	}
```

En aquest cas, com hi ha una classe definida exclusivament per a les consultes, el FormBean només té dos mètodes, un per a la recuperació i l'altre per a la introducció de la consulta.

**ConsultaControl.java**

```
package control;
import java.sql.Date;
import model.Consulta;
import org.apache.beehive.controls.api.bean.*;

@ControlInterface
public interface ConsultaControl
{
	public boolean afegirConsulta(Consulta consulta);

	//...

}
```

**ConsultaImpl.java**

```
package control;
import java.io.Serializable;
import java.sql.Date;
import java.util.Calendar;
import model.Consulta;
import org.apache.beehive.controls.api.bean.*;

@ControlImplementation
public class ConsultaControlImpl implements ConsultaControl, Serializable {
	private static final long serialVersionUID = 1L;

	@Control()
	private ConsultaDB consultaDB;

	public boolean afegirConsulta(Consulta consultilla)
	{
		try
		{
			int ID = consultaDB.getConsultaID()+1;
			consultilla.setIdConsulta(ID);
			int Any = Calendar.getInstance().get(Calendar.YEAR);
			int Tancat = consultaDB.ComprovarAny();
			if (Tancat == Any)
```

```
                    consultilla.setCodi((Any+1) + "/" + ID);
                else
                    consultilla.setCodi(Any + "/" + ID);
                consultaDB.afegirConsulta(consultilla);
                return true;
            }
            catch(Exception e)
            {
                if(e.getCause()!=null)
                    System.out.println(e.getCause().toString() +  ". Error en BBDD. No s'ha guardat la consulta.");
                else
                    System.out.println(e.toString() + ". Error en BBDD. No s'ha guardat la consulta.");
                return false;
            }
        }

        //...
}
```

Dintre ConsultaControlImpl.java es pot veure com el mètode afegirConsulta realitza aquesta adició a la base de dades. Primer es recupera l'última ID, el camp numèric que indexa les consultes, per generar el codi, que consta de l'any i la ID. En funció si l'any s'ha tancat o no, s'utilitza una dada o l'altra. Un cop generat aquest codi, s'adjunta a la Consulta i aquesta s'envia al mètode contingut en ConsultaDB.java, que crea la sentencia SQL que crearà el nou registre amb la consulta validada dintre la taula Consultes dintre la base de dades.

**ConsultaDB.java**

```
package control;

import model.Consulta;
import java.sql.Date;
import org.apache.beehive.controls.api.bean.*;
import org.apache.beehive.controls.system.jdbc.*;

@ControlExtension
@JdbcControl.ConnectionDataSource(jndiName = "oracle")
public interface ConsultaDB extends JdbcControl {

        //Afegir Nova Consulta
        @JdbcControl.SQL(statement = "INSERT INTO CONSULTA" +" ( IDCONSULTA, NOM, COGNOMS, CORREU, TELEFON,
DATACOMUNICACIO, USUARIRESPONSABLE," +" TIPOLOGIA, CONSULTA, SOLUCIOFINAL, ESTAT, DATACONFORME, CODI,
UNITATADMINISTRATIVA )" +"  VALUES " +"(TO_NUMBER({cons.IdConsulta}), {cons.nom},{cons.cognoms},  {cons.Correu}," + "
{cons.Telefon}, TO_DATE({cons.DataComunicacio},'dd-Mon-yyyy HH:MI:SS AM'), {cons.usuariResponsable}," + " {cons.tipologia},
{cons.Consulta}, " + " {cons.SolucioFinal}, {cons.Estat}, TO_DATE({cons.DataConforme}), {cons.codi}, {cons.unitatAdministrativa}) " )
        public void afegirConsulta(Consulta cons);

        //...
}
```

La implementació de la sentència SQL és relativament senzilla, ja que com es pot veure al fitxer ConsultaDB.java, es limita a la codificació del codi, deixant la configuració de la connexió JDBC al domini del servidor Web.  La cadena SQL agafarà les dades del objecte passat per paràmetre al mètode, i aquesta s'enviarà al gestor de bases de dades, el TNS Listener de Oracle. Aquest  rebrà la petició SQL provinent del servidor Weblogic, juntament amb les dades de connexió a la base de dades Uhelpdesk: IP, Port, Nom d'usuari, Password i SID, que les proporciona el domini, com ja s'ha dit. Un cop acceptada i executada la sentència, es tancarà la connexió del aplicatiu a la base de dades, i el servidor d'aplicacions continuarà executant el mètode. Dintre de ConsultaControlImpl.java, si la operació s'ha completat retorna un true. Aquest true es recollit per la funció if en el mètode AfegirConsulta del controlador de flux. Com que s'ha retornat un true, el intèrpret seguirà executant el codi immediatament consecutiu a la comparació. Aquest es tracta de la creació del objecte Forward, que conté l'adreça de destí, i els paràmetres nadius de Beehive que seran interpretats per la pàgina JSP. En la implementació mostrada en Pf2Controller.java, es pot veure com es crea una matriu de consultes, i que en la primera posició es guarda la que s'acaba de guardar. Això és així per que la pàgina de destí es compartida amb l'acció de buscar consultes, i per tant, la pàgina està codificada

per que rebi més d'una consulta com a paràmetre d'entrada. Per a tal, quan afegim la consulta, la fiquem dintre una matriu buida i l'enviem a la pàgina ConsultaResultat.jsp. Aquesta pàgina conté un llistat de les consultes que li són passades, i en el nostre cas, permet a l'usuari comprovar que les dades introduïdes són correctes.

**Cerca de consultes:** Per a la cerca de consultes, s'empra un mètode únic per a tots els criteris, amb el qual es passen els paràmetres possibles per fer la cerca, i aquests són gestionats dintre el mètode, cridat des de la pàgina Consultes.jsp. D'aquesta manera s'aprofita el codi per a cada cas en que sigui necessari obtenir una consulta a partir de una o diverses dades. Els paràmetres d'aquest mètode són els següents: Identificació de consulta, NIF, Nom, Cognoms, Servei, Secció i Estat. En la implementació del mètode, es comprova el contingut de cadascun dels paràmetres, tancant les possibles combinacions i utilitzant només un mètode de ConsultaControlImpl.java. Per a la cerca s'utilitzen diferents consultes SQL en funció dels criteris de cerca, demanant els objectes Consulta, que seran retornat dintre una matriu de consultes, per ser representades en la pàgina ConsultaResultat.jsp

**Modificació de consultes:** Per a la modificació de les consultes, de la mateixa manera que en modificar qualsevol altre registre de l'aplicatiu, s'accedeix a una pàgina formulari mitjançant un enllaç existent en la pàgina Web de resultats Al carregar-se la pàgina el formulari s'emplena de les dades que eren contingudes a la base de dades. L'usuari només té que modificar les dades desitjades. Quan es valida el formulari, l'aplicatiu recull el formulari, i envia la nova consulta a la base de dades per substituir les noves dades per les introduïdes utilitzant com a referència el camp ID, que identifica la consulta . La sentència emprada per a la modificació de les dades és UPDATE.

**Eliminació de consultes :** Per a la eliminació de les consultes, es recorre a un enllaç desplegat en els resultats de la consulta dintre ConsultaResultat.jsp. En aquest cas, aquest enllaç apunta a una pàgina Web que ens demana la confirmació per a eliminar el registre en concret, a través d'un senzill formulari on es disposen dos botons, un que activa l'acció d'eliminar i l'altre que ens torna enrere. Si premem el botó amb el text "Sí", l'acció executa el mètode que crida la sentència SQL DELETE a la base de dades per enviar-li l'ordre d'enviar el registre juntament amb el valor del codi igual al que s'ha passat per paràmetre.

**Promoció de consultes a incidències o a peticions:** En el cas de les consultes, en la pàgina de resultats, juntament amb els enllaços de modificació i eliminació, es disposen enllaços que permeten promocionar la consulta a incidència o bé a petició. Amb aquest enllaç, es passa a un formulari específic que està preparat per a les dades de incidència o bé de petició, i es emplenat amb les dades provinents de la consulta existent. L'usuari té que afegir les dades que manquen per completar el registre de la incidència/petició. Si valida el formulari, es realitzen les dues accions necessàries, o sigui, s'elimina la consulta promocionada, i s'afegeix una nova incidència/petició a la base de dades, amb la informació provinent del formulari. Aquesta promoció a més, comporta la generació de l'informe corresponent i el seu enviament per mitjà de correu electrònic a la bústia que s'ha facilitat en el formulari. Per completar aquest enviament és necessari que les dades SMTP entrades a la secció d'Opcions siguin correctes.

**Noves Intervencions:** Per a l'addició de noves intervencions, tant per incidències com per peticions, es recorre a un enllaç en forma de botó, dintre del formulari de modificació. D'aquesta manera es separa la consulta de dades, de la seva modificació. Amb aquest botó, accedim a un formulari específic per a Intervencions. Un cop hem emplenat les dades del formulari, i hem validat el mateix, aquest s'afegeix a la base de dades, en una taula separada de la incidència mare, però que està referenciada a la mateixa a través d'un camp específic anomenat IdIncidència. En el cas de Intervencions provinents de Peticions, la mecànica és idèntica.

**Autenticació dels usuaris dintre l'aplicatiu :** Per a l'autenticació dels usuaris en l'aplicatiu, es prescindeix de les capacitats intrínseques de J2EE (javax.security), i s'implementa una autenticació simple a través de la base de dades.

**Navegació de l'usuari per l'aplicatiu :** Per a la navegació entre pàgines, es genera una acció simple, anomenada dintre Struts com a ...

## *5.5 Descripció de les classes auxiliars*

**Àrees:** Aquestes classes estan implementades senzillament per un conjunt de cadenes de text que permeten guardar les dades de l'àrea administrativa que es correspon amb la secció orgànica de l'organització. Els mètodes que es defineixen per a aquesta classe són els elementals per a una classe com aquesta : RetornaArees() que no necessita de paràmetres, i retorna una matriu de cadenes de text, que pot ser emprada per emplenar una llista desplegable per a un formulari. A més, s'implementen els mètodes NovaArea i EsborraArea, que com el seu nom indiquen permeten introduir i esborrar de la base de dades àrees pertanyents als departaments de la organització. El primer mètode té com a paràmetre la pròpia classe Area, de manera que aquesta es trasllada a la base de dades, mentre que la segona, només té el valor del ID com a valor d'entrada, que s'usa com a referència al hora d'esborrar la àrea.

**Responsables Àrees, Tipologies, Estats, Unitats Administratives, Responsables Unitats Administratives:** Aquestes classes de tipus genèrics es poden resumir com a contenidors de cadenes de text, que generalment s'utilitzen per caracteritzar les consultes o incidències. En el cas dels Estats, s'utilitza per realitzar cerques dintre la base de dades. Totes comparteixen els mètodes bàsics per a la seva gestió : creació, modificació i eliminació. La creació és realitza mitjançant un formulari que s'emplena amb les dades necessàries, com per exemple, el nom que es mostrarà en els llistats, i la seva validació comporta la introducció en la base de dades, i la seva aparició en les llistes desplegables que podem trobar als formularis de introducció de dades. Per altra banda, els mètodes d'eliminació fan referència a través de la seva ID.

**Usuaris:** La classe Usuari caracteritza els usuaris de Helpdesk. Conté les dades d'identificació i de autenticació del usuari. Aquest nom d'usuari serà introduït en qualsevol incidència introduïda al sistema, de manera que es pugui localitzar per l'usuari responsable, terme que apareix en els formularis de cerca. Els usuaris es classifiquen es dos perfils diferents : gestor i consultor, els quals es diferencien per la seva capacitat de modificar les dades existents, i usuaris associats a una Unitat Territorial, o bé amb caràcter omnipresent, que té accés a les dades de qualsevol Unitat Territorial.

**Tancar Any:** Aquest mètode s'utilitza per establir una data la qual totes les incidències anteriors, no seran llistades i romandran inaccessibles. Únicament seran comptades a efectes de Informes Anuals. Aquest mètode introdueix la data a la base de dades, i el sistema, en cada cerca, va comprovant la data de comunicació de la incidència, que serveix com a referència en aquest cas.

## *5.6 Base de dades*

La base de dades realitza les gestions d'emmagatzemar la informació que se li passa i retornar aquella que s'ajusta a les consultes que se li fan. Aquest s'emmagatzema en taules, de manera que podem dissenyar un entorn regit per columnes, que seran els que definiran cada element del conjunt de dades on anirà guardat i aquests conjunts de dades s'agrupen en files, formant un full de dades que és la forma que millor representa l'emmagatzematge de dades.

En el cas d'aquest aplicatiu, la configuració del servidor de bases de dades Oracle es basa en la configuració més senzilla: un servidor, una base de dades i un usuari que realitza insercions, consultes, modificacions i esborraments. Per tant, la implementació es resumeix a la creació de la base de dades i de les seves taules, en la configuració del servidor i en la creació d'un usuari que permetria l'accés a la base de dades.

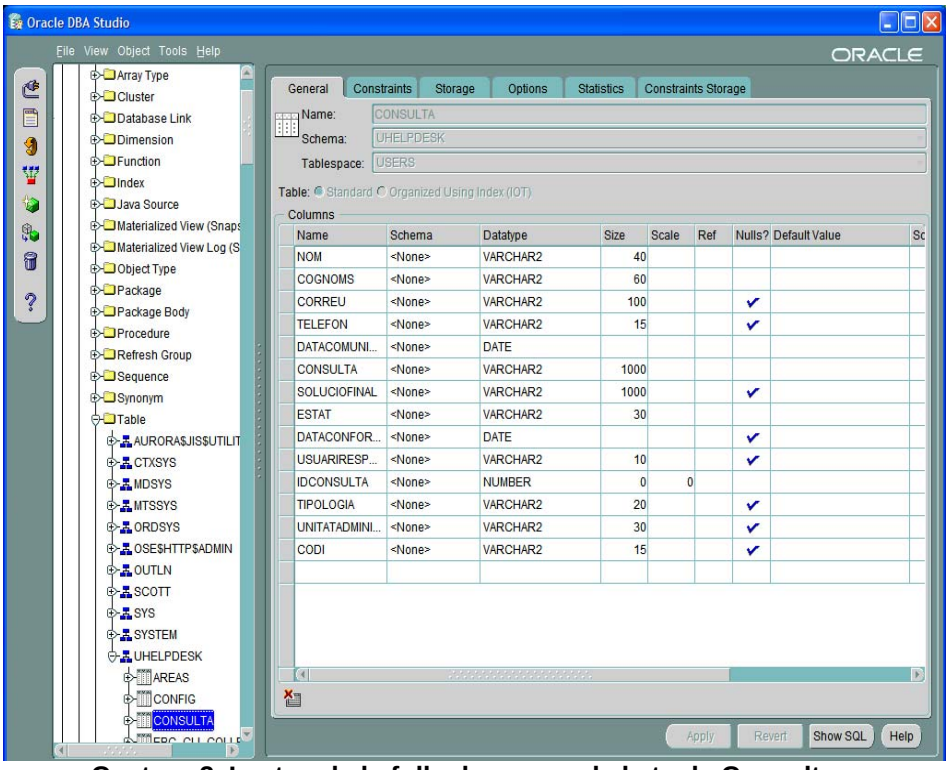
Primer, es crea la base de dades, i després, es defineix com guardar la informació en taules, l'estructuració és la següent:

- Àrees: Taula destinada a memoritzar les àrees de intervenció que componen el servei d'informàtica.
- Config: Taula auxiliar que enregistra dades utilitzades en el funcionament de l'aplicatiu.
- Consulta: Taula on s'enregistren les dades sobre les consultes.
- Estats: Taula on es guarden els estats possibles.
- Incidència: Taula on s'enregistren les dades sobre les incidències.
- Intervencions: Taula on s'enregistren les intervencions.
- Notes: Taula per a les notes personals.
- Peticions: Taula on s'enregistren les dades sobre les peticions
- Responsables Àrees: Taula on queden enregistrats les dades dels Responsables de Àrea.
- Responsables Unitats Administratives: Taula on s'enregistren els Responsables d'Unitats Administratives (UA)
- Tipologies: Taula amb els tipus de consultes aplicables.
- Unitats Administratives: Llistat de UA
- Usuaris: Taula amb les dades dels usuaris.

Les característiques comunes de les taules, en temps de disseny, són el camp ID amb l'objectiu de recuperar informació de varies taules de cara al usuari.

L'altre camp important i comú en la majoria de camps que representen una entitat, és la DATA, que ens permet tenir una referència temporal de qualsevol entitat que sigui interessada per ser reflectida en els informes.

La resta de camps s'han creat seguint l'exemple de l'aplicatiu anterior, afegint aquells que incorporen noves funcionalitats, com ara, el camp que descriu l'usuari responsable, que permet saber quina persona de Helpdesk ha fet el seguiment de la incidència. En el següent esquema es veuen reflectides les taules que es van enumerar en un primer moment, el nom del camps que contenen, a més del tipus de dades, longituds, si poden o no tenir un valor nul, el valor inicial que els hi ha d'assignar la base de dades en el moment de crear el registre, i una breu descripció de cadascun dels camps. En la següent captura podem veure la configuració de la taula Consulta, amb els seus camps**:**




**Captura 2. Imatge de la fulla de camps de la taula Consultes.**

## *5.7 Pàgines*

Per implementar les pàgines web d'acord amb la estructura lògica dels aplicatius de gestió d'informació, aquest començarà amb una pàgina d'autenticació. A partir d'aquesta, accedirem a una pàgina amb un menú principal des del qual podrem accedir a les principals parts: Consultes, Incidències i Peticions per un costat, Informes per un altre i finalment un subgrup, com ara, Opcions. Aquesta separació en funció de la naturalesa de la incidència sembla la més lògica, ja que podem gestionar qualsevol incidència com a consulta, per després, en funció del seu anàlisis, promocionar-la a incidència o petició. Per altra banda, els informes i les opcions són considerades com a opcions principals de l'aplicatiu, pel fet de tenir moltes opcions per a la seva visualització/configuració. Des d'un punt de vista general, la classificació dels que serien les pàgines/formularis, on es desenvoluparia l'activitat funcional, és aquesta:

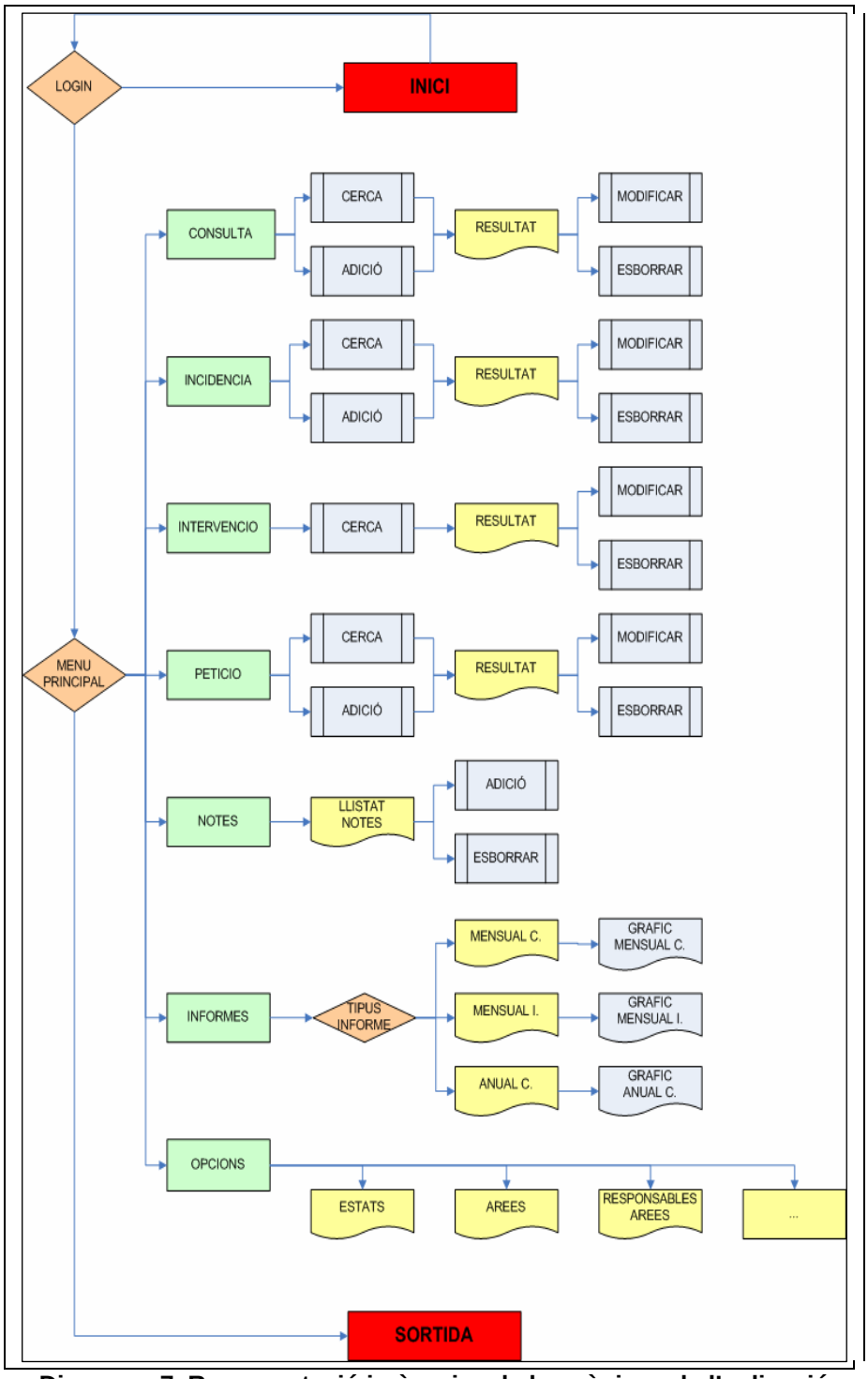



**Diagrama 7. Representació jeràrquica de les pàgines de l'aplicació**

Analitzant aquesta distribució, podem destacar com avantatge que no hi ha cap secció que estigui a més de dos clics. El menú principal és estàtic, i tot i no ser configurable, és clarament identificable i de aprenentatge immediat, de manera que no s'ha d'accedir a diferents llocs per fer la mateixa acció.

A l'hora de convertir les dades que arriben al servidor en forma de petició HTTP, s'empra el que el s'anomena Form Bean, o bean de formulari. Un bean, entitat de codi, amb les seves propietats i mètodes, es crea dinàmicament per BEA Workshop per tal de traslladar la informació provinent de les etiquetes HTML a les propietats de la classe desitjada, a través de la construcció del bean, el seu

“emplenament”, i posteriorment la seva destrucció. El següent codi pertany a una pàgina-formulari d'entrada de dades per a la constitució de una consulta :

**Consultes.jsp**

```
<%@ page language="java" contentType="text/html;charset=UTF-8"%>
<%@taglib uri="http://beehive.apache.org/netui/tags-html-1.0" prefix="netui"%>
<%@taglib uri="http://beehive.apache.org/netui/tags-databinding-1.0" prefix="netui-data"%>
<%@taglib uri="http://beehive.apache.org/netui/tags-template-1.0" prefix="netui-template"%>
<netui-data:declarePageInput required="false" type="java.lang.String" name="Missatge" />
<netui-data:declarePageInput required="true" type="String[]" name="_Usuaris" />
<netui-data:declarePageInput required="true" type="String[]" name="_EstatsTots" />
<netui-data:declarePageInput required="true" type="String[]" name="_EstatsOberts" />
<netui-data:declarePageInput required="true" type="String[]" name="_UA" />
<netui-data:declarePageInput required="true" type="String[]" name="UsuariActiu" />
<netui-data:declarePageInput required="false" type="String[]" name="_Tipologies" />
<link rel="stylesheet" href="img/primer.css" type="text/css">
<netui:html>
<head>
	<netui:base/>
</head>
<netui:body>
<%@ include file="Marc_Superior.jsp" %>
<%@ include file="Menu2.jsp" %>
<p>Consultes</p>
<div id = "Missatge">${pageInput.Missatge }</div>
<br>
<b>BUSCAR CONSULTA</b>
<netui:form action="BuscadorConsultes">
		<div>
		<table>
		<tr valign="top">
			<td><label for="field1"> Codi : </label></td>
			<td><netui:textBox dataSource="actionForm._Codi" tagId="field1"></netui:textBox></td>
		</tr>
		<tr valign="top">
			<td><label for="field2"> Responsable : </label></td>
			<td><netui:select dataSource="actionForm._Responsable" tagId="field2" optionsDataSource="${pageInput._Usuaris}"
nullable="true" nullableOptionText="---" ></netui:select></td>
		</tr>
		<tr valign="top">
			<td><label for="field3"> Cognoms : </label></td>
			<td><netui:textBox dataSource="actionForm._Cognoms" tagId="field3"></netui:textBox></td>
		</tr>
		<tr valign="top">
			<td><label for="field4"> Estat : </label></td>
			<td><netui:select dataSource="actionForm._Estat" tagId="field4" optionsDataSource="${pageInput._EstatsTots}"
nullable="true" nullableOptionText="---"></netui:select></td>
		</tr>
		<tr valign="top">
			<td><label for="field5"> Data de Comunicació (dd/mm/aaaa) : </label></td>
			<td><netui:textBox dataSource="actionForm._Data" tagId="field5"><netui:formatDate pattern="dd/MM/yyyy"
stringInputPattern="dd/MM/yyyy" /></netui:textBox></td>
		</tr>
		<tr valign="top">
			<td><label for="field6"> Unitat Administrativa : </label></td>
			<td><netui:select dataSource="actionForm.UA" tagId="field6" optionsDataSource="${pageInput._UA}" nullable="true"
nullableOptionText="---" nullable="true" ></netui:select></td>
		</tr>
		</table>
		</div>
		<netui:imageButton src="./img/imgc7.gif" rolloverImage="./img/imgo7.gif"></netui:imageButton>
</netui:form>

<b>AFEGIR CONSULTA</b>

<netui:form action="afegirConsulta">
		<div>
		<table>
		<tr valign="top">
			<td><label for="field1"> Estat : </label></td>
```

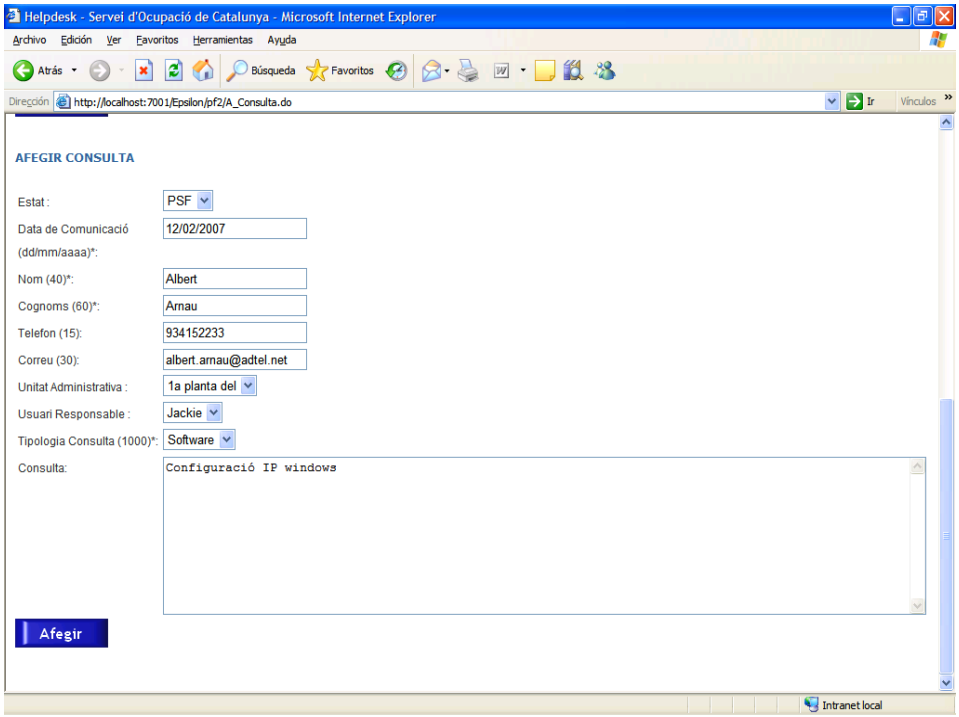
```
                <td><netui:select dataSource="actionForm.consulta.estat" tagId="field1"
optionsDataSource="${pageInput._EstatsOberts}"></netui:select></td>
        </tr>
        <tr valign="top">
                <td><label for="field2"> Data de Comunicació (dd/mm/aaaa)*: </label></td>
                <td><netui:textBox dataSource="actionForm.consulta.dataComunicacio" tagId="field2">
                <netui:formatDate pattern="dd/MM/yyyy" stringInputPattern="dd/MM/yyyy" /></netui:textBox></td>
        </tr>
        <tr valign="top">
                <td><label for="field3"> Nom (40)*: </label></td>
                <td><netui:textBox dataSource="actionForm.consulta.nom" tagId="field3"></netui:textBox></td>
        </tr>
        <tr valign="top">
                <td><label for="field4"> Cognoms (60)*: </label></td>
                <td><netui:textBox dataSource="actionForm.consulta.cognoms" tagId="field4"></netui:textBox></td>
        </tr>
        <tr valign="top">
                <td><label for="field8"> Telefon (15): </label></td>
                <td><netui:textBox dataSource="actionForm.consulta.telefon" tagId="field8"></netui:textBox></td>
        </tr>
        <tr valign="top">
                <td><label for="field9"> Correu (30): </label></td>
                <td><netui:textBox dataSource="actionForm.consulta.correu" tagId="field9"></netui:textBox></td>
        </tr>
        <tr valign="top">
                <td><label for="field10"> Unitat Administrativa : </label></td>
                <td><netui:select dataSource="actionForm.consulta.unitatAdministrativa" tagId="field10"
optionsDataSource="${pageInput._UA}"></netui:select></td>
        </tr>
        <tr valign="top">
                <td><label for="field5"> Usuari Responsable : </label></td>
                <td><netui:select dataSource="actionForm.consulta.usuariResponsable" tagId="field5"
optionsDataSource="${pageInput._Usuaris}" defaultValue="${pageInput.UsuariActiu}"></netui:select></td>
        </tr>
        <tr valign="top">
                <td><label for="field7"> Tipologia Consulta (1000)*: </label></td>
                <td><netui:select dataSource="actionForm.consulta.tipologia" tagId="field5"
optionsDataSource="${pageInput._Tipologies}"></netui:select></td>
        </tr>
        <tr valign="top">
                <td><label for="field6"> Consulta: </label></td>
                <td><netui:textArea dataSource="actionForm.consulta.consulta" tagId="field6" cols ="100"
rows="10"></netui:textArea></td>
        </tr>
        </table>
        </div>
        <netui:imageButton src="./img/imgc8.gif" rolloverImage="./img/imgo8.gif"></netui:imageButton>
</netui:form>

</netui:body>
</netui:html>
```

En el codi JSP de la pàgina Consultes.jsp, podem veure que conté dos formularis: el primer cerca les consultes, en funció de diversos paràmetres, com poden ser la Data de comunicació, l'usuari responsable de la consulta, l'estat en que es troba, etc. Per altra banda, l'altre formulari està dedicat a la introducció de noves consultes, amb elements d'entrada de dades. En aquesta pàgina el que podem destacar són l'ús de Selects, que recullen les opcions dels pageInputs definits al principi de la pàgina i l'ús d'etiquetes que ens permeten que les dades recuperades, siguin formatades, en el nostra cas en forma de data.

La personalització de **l'entorn d'usuari** a l'hora de crear les pàgines Web. Això es va solucionar mitjançant l'ús de CSS (Fulles d'estils en cascada). Per al desenvolupament de aquests estils va ser necessari una aplicació. Es va escollir una que també fos lliure. Per a tal es va escollir EclipseStyle, un aplicatiu senzill, i eficient. Es basa en la modificació del fitxer CSS a partir de text, a la vegada que es disposa d'un formulari per configurar automàticament l'estil desitjat. A més, és WYSIWYG, de manera que segons anem modificant les propietat, un text de mostra com es van aplicant els canvis.

Observem el codi emprat per la capa d'abstracció de dades perquè introdueix el prefix netui a les sentencies. Aquestes etiquetes, alhora de ser rebudes pel navegador es substituiran per sentencies HTML equivalents. Aquesta abstracció facilita la tasca del programador al introduir una capa intermediària entre HTML necessari per a la representació del codi Java de l'aplicatiu. L'altra observació a fer és el cas en que les dades fan el camí invers. En aquest formulari hi han dades que són demanades al servidor, per a poder ser representades dintre el codi HTML. Fixar-se en el cas de les etiquetes netui:select que corresponen a camps de selecció (Servei i Secció), en que l'usuari escull entre unes dades ja determinades. El següent codi pertany a un mètode que creem dintre el fitxer controlador de flux que caracteritza l'enviament del formulari anterior.

En la següent captura es pot veure el formulari de introducció de consultes:




**Captura 3. Captura de pantalla de la pàgina de Consultes.jsp**

# 6 REPTES EN EL DESENVOLUPAMENT DE L'APLICATIU

Durant el desenvolupament de l'aplicatiu, s'han presentats diferents reptes de diferent nivell. Destacaria el fet de utilitzar un Servlet extern per a la representació gràfica dels informes

La **generació d' informes i gràfics** per l'anàlisi dels informes d'activitats del Servei de Helpdesk. Per a desenvolupar gràfics i informes sobre Java existeixen diverses solucions en forma de llibreries o Servlet que funcionarien conjuntament amb l'aplicatiu. Es va buscar una solució que fos software lliure, o sigui, amb drets de còpia i utilització. De les llibreries més destacades estan les proporcionades per Crystal Reports, que és un referent gràcies a la seva experiència en informes sobre sistemes Windows i escollir entre la gran oferta d'aplicacions que implementen gràfiques i que són Open Source.

Per realitzar aquesta tasca, es va avaluar les despeses de temps en desenvolupar un propi sistema de representació gràfica, basat en llibreries natives de Java en 2D. Però també es va estudiar la possibilitat d'emprar un conjunt de classes que permetis guanyar temps, amb uns resultats millors. Entre moltes possibilitats es va seleccionar una llibreria de llicència de franc anomenada JFreeChart.

Aquesta llibreria permetia el lliure ús i distribució, però la documentació era remunerada, però els exemples compilats, ni els codis fonts del mateix. A partir d'un exemple es van delimitar les possibilitats de la llibreria i es va poder seleccionar les característiques més adients per al funcionament amb l'aplicatiu.

Concretament, es va seleccionar un model de gràfic que es basa en una sèrie temporal de valors. En l'eix de les X es representa l'escala temporal i en les Y els valors. Aquestes sèries venen definides per una classe, que en herència disposa de classes per unitat de temps, és a dir: segon, minut, hora, dia, mes i any. Per tant, la realització del gràfic és limita a la creació de les escales X i Y, al fons, i a la creació de les sèries, que eren estàtiques en funció del tipus de gràfic. Però el que va resultat més interessant era trobar la manera de cridar aquest Servlet des del Servlet de l'aplicatiu, passant com a paràmetres les col·leccions de entitats (recordem: Consultes, Incidències, etc.). El llenguatge Java permet aquesta operació, però el repte era trobar la URL amb la qual el Servlet destí fos capaç, primer, d'escoltar la petició, i després, de convertir les dades passades com a paràmetre.

La generació i **enviament automàtic del missatge de correu electrònic** juntament amb la formació d'un **document en format RTF que** és necessari adjuntar juntament amb els correus electrònics. Aquest document respon a un informe de consulta, que conté un text predeterminat. Per a la gestió d'aquest tipus de fitxer es va cercar una sèrie de solucions. Es basa en la incorporació al projecte, d'una llibreria que proporciona una sèrie de classes que permeten generar un document RTF mitjançant un parser. Aquesta llibreria retorna una classe File, el qual el seu contingut és un document RTF. Servidor XML2PDF. Es basa en un servei SUN contra el qual podem enviar peticions de conversió en format XML. En funció de la petició podem demanar que se'ns retorni un document RTF, dintre un conjunt de bytes, la qual podem incorporar dintre una MimeBodyPart i enviar-lo com a adjunt al correu electrònic. Gestió a nivell de caràcter. Una opció és utilitzar un RTF de mostra, el qual conté una sèrie de caràcters en funció de tokens, i modificar aquests tokens per introduir les dades corresponents. Un cop modificat, enviar la cadena de bytes a través del constructor de la MimeBodyPart.

El format del informe, a petició del client, és Rich Text Format (RTF). El client va proporcionar una mostra per poder crear un model i aquest adjuntar-lo com a Adjunt. Per a tal, no es va recórrer a cap classe o llibreria externa, degut a que el text RTF està basat en text pla, i la classe Javamail ja implementa qualsevol necessitat que pugui requerir un correu electrònic.

# 7 JUSTIFICACIÓ DE LES SOLUCIONS ESCOLLIDES

**Informes i generació de gràfics:** Quan accedim a la pàgina d'Informes, el primer que fa l'aplicatiu és buscar entre les incidències, la entrada més antiga, mitjançant una sentència SQL. Amb les tres dates provinents de les tres taules, podem seleccionar la més antiga, mitjançant un algorisme d'ordenació, amb tres comparacions.

El següent és generar una matriu de dates de manera que el període de temps entre la primera data i el moment actual. Aquest algorisme l'implementem gràcies al objecte Calendari. El Calendari permet canviar el dia, el mes i l'any sobre valors enters, és a dir, poder sumar-li un any, o restar-li 3 dies, amb el que el Calendari ajustarà les dates, tenint en compte, fins i tot, els anys de traspàs. Amb aquesta eina, anirem recorrent un Calendari, amb data original igual a la primera, mes a mes, creant una matriu de mesos, i cada dotze mesos, una matriu de anys. Amb aquestes dues matrius, generarem dues matrius de cadenes de text per ser representades per pantalla. Amb aquests dos paràmetres, es mostra la pàgina d'Informes.
A l'hora de seleccionar l'informe, s'agafa la data seleccionada, a quadre proper al botó corresponent a l'informe.

La següent acció es pràcticament idèntica en els tres casos. Quan l'acció recupera per paràmetre la data que s'ha seleccionat al fer clic en el botó, es converteix a Calendari. Aquest calendari, que es correspon amb la data que representa el període pel qual es volen buscar incidències, el desplacem un mes endavant, i generem una segona data. Amb aquesta data tenim el que necessitarem per cercar les incidències dintre la base de dades, ja que els mètodes de cerca construeixen la sentència SQL en funció de les dues dates, ja que demanarem a la base de dades les incidències les quals la data de comunicació sigui superior a la primer (dia 1) i inferior a la segona (últim dia del mes). El resultat d'aquestes sentències són portades directament a la següent pàgina.

Els informes anuals recullen l'activitat registrada en l'any natural. Per a efectuar aquesta selecció de registres dintre la base de dades, s'implementa amb la base de que la pàgina JSP, rep tres matrius, una de cada tipus d'incidència. Amb aquesta premissa, es va implementar el mètode que permeti oferir la informació en pantalla. Per a tal, l'algorisme comença amb la cerca de la que seria la primera incidència. Fa una crida a la base de dades per a que li retorni la data més antiga, amb aquesta dada, es van recuperant incidències dintre d'aquest període. Es presenten per pantalla, i per la generació de gràfics, s'utilitza una sentencia que centra la cerca dintre un període de temps determinat.

La generació del gràfic es realitza dintre un Servlet nou, seguint l'exemple proporcionat per JFree.org. L'acció de generar el gràfic fa la petició HTTP al Servlet, i li adjunta com a atributs les matrius d'incidències, més una cadena de text amb el qual s'identifica el gràfic desitjat.

Aquests atributs són recollits pel server, introduïts a les classes adients i llavors es traslladen al procés de creació del gràfic. Hem ajustat aquest per ser un gràfic de línies, amb els títols personalitzats, sobre una carta de coordenades dinàmica, amb els intervals de temps a les ordenades i el nombre d'incidències a les abscisses. Per a la generació d'un gràfic de línies, JFreeChart es basa en els següents passos :

Primer que res es necessari capturar la petició. S'ha implementat captura tant per al GET i per al POST, fent que tots dos criden al mateix mètode, passant com a paràmetres la petició, amb atribuïts inclosos, i la resposta, per a que el mètode la empleni amb el gràfic.

Un cop es processa la resposta, el primer que es fa es establir el tipus de resposta, que entre les possibilitats que teníem (PNG,JPG), hem escollit JPG. Després es creen els dos objectes necessaris per generar el gràfic : El XYDataset, específic per a gràfics de línies en 2D, i l'objecte JFreeChart que

representa el gràfic en sí mateix. Tot seguit es selecciona el camí en funció del tipus de gràfic. Cadascun les mètodes possibles, tenen en comú la creació del Dataset, la implementació del gràfic en funció d'aquest Dataset, la unió del stream de dades entre la sortida del gràfic i la resposta HTTP, un cop aquesta, el inici de la creació del gràfic, de manera que alhora s'empleni la resposta, i al finalitzar, el tancament d'aquest Stream, per poder soltar la resposta HTTP.

El primer pas, la creació del Dataset es basa en la classe TimeSeries que s'ofereix amb la llibreria JFreeChart. Aquesta classe conté les propietats i el mètodes necessaris per relacionar unitats de temps amb un comptador, el qual permet representar gràficament una progressió temporal. Per això s'ha implementat un bucle que en primer terme, recorre el període de temps, i en segon terme, recorre les matrius, fins al final, ja que aquestes ja estan filtrades per la DataComunicació. Per al registre de diferents conceptes, s'han de crear diferents TimeSeries. Per exemple, tenim dos TimeSeries per registrar les Incidències de primer i segon grau. I per implementar aquest bucle, s'han creat dos Calendaris per establir un període de temps amb el que comparar la data de comunicació. Mitjançant aquest període podem ubicar cada unitat de la matriu en un període de temps, de manera que es pot afegir al TimeSeries inequívocament.

**Gestió del correu electrònic:** S'ha implementat un mètode amb l'ajuda de la llibreria interna JavaMail. Quan es genera una Incidència o bé una Petició, es va requerir l'enviament d'un informe a l'usuari afectat. Aquest informe ve formatat dintre un fitxer RTF. Quan es genera el correu electrònic, abans de ser enviat, s'afegeix una MultiPart, que corresponen al RTF de l'informe. Per construir l'informe RTF ens hem basat amb una plantilla feta, que incorpori les tabulacions i el format del text. Mitjançant l'edició del mateix, es recorre el fitxer en busca de una sèrie de paraules clau que corresponen amb els camps de la Incidència/Petició, per després ser enviat. Per a implementar aquesta funcionalitat, s'ha deixat una còpia de la plantilla en la carpeta arrel del servidor : Aquest el recupera, i el va llegint línia per línia dintre un String. Quan s'ha introduït tota la plantilla, llavors s'utilitza la funció Replace per intercanviar les paraules claus pels valors de les classes Incidència/Petició. Un cop finalitzada la substitució, es crea un DataHandler, a partir de l'espai ocupat per la cadena de text i dels propis caràcters que permeten la creació d'un objecte que es apte per la incorporació al correu electrònic en forma de MultiPart.

La classe Correu, conté les propietats mínimes necessàries per caracteritzar un correu e-mail senzill, és a dir, la definició del destinatari, remitent, subjecte, cos i adjunts. A més, per a la gestió de les opcions del servidor de correu, i la implementació dels mètodes que executaran l'enviament, la recepció, etc. es va realitzar en un conjunt de classes. La classe Opcionscorreu contenia informació més relativa a la configuració del servei de correu de l'entorn, és a dir, nom del servidor SMTP, port, etc. Es tracta d'una classe on es guarden les propietats que conformen la informació necessària per connectar-se al servidor de correu i enviar un correu electrònic que està contingut dintre de la classe Correu. Per enviar un correu senzill, amb l'ajut de JavaMail, s'utilitza el següent codi d'exemple, que es el que es va utilitzar com a base per desenvolupar el mètode que es crida dintre l'aplicatiu:

```
String host = ...;
String from = ...;
String to = ...;

//  Recuperació de les propietats del sistema
Properties props = System.getProperties();

// Posta a punt del servidor de correu
props.put("mail.smtp.host", host);

// Creem la sessio a partir de les propietats
Session session = Session.getDefaultInstance(props, null);

// Definim les característiques del missatge
MimeMessage message = new MimeMessage(session);
message.setFrom(new InternetAddress(from));
message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
message.setSubject("Hello JavaMail");
```

```
message.setText("Welcome to JavaMail");

// Finalment enviem el missatge.
Transport.send(message);
```

A partir d'aquesta base, es va anar desenvolupant aquest servei de correu. La primera cosa que es va fer, va ser crear la classe Correu, per que el mètode d'enviament requerís el mínim nombre de paràmetres. Aquesta classe correu consta de les següents propietats :

```
private String Host;
private String Remitent;
private String Destinatari;
private String Assumpte;
private String Cos;
private String CC;
private boolean isHTML;
private boolean debug;
private org.apache.struts.upload.FormFile AdjuntFormFile;
private String Adjunt;
private int index;
private Date data;
private MimeBodyPart RTF;
```

En el primer cas, la generació de gràfics es va realitzar a través del següent codi. Per la part del controlador de flux cap al Servlet de gràfics:

Generació de gràfics de consultes, incidències i peticions anuals:

```
@Jpf.Action(forwards = {
                        @Jpf.Forward(name = "success", path = "../fes.graficAtt"),
                        @Jpf.Forward(name = "exit", action = "Logout")})
public Forward GRAFICAC(graficac form)          {
        if (getusuari() == null)
                return new Forward("exit");
        HttpServletRequest req = getRequest();
        Forward forward = new Forward("success");
        Date d = form.getDataA();
        Consulta[] cons = consultaControl.RetornaInforme(d);
        Incidencia[] insi = incidenciaControl.retornaInforme(d);
        Peticio [] pet = peticioControl.retornaInforme(d);
        Intervencio[] interI = intervencioControl.retornaInformeI(d);
        Intervencio[] interP = intervencioControl.retornaInformeP(d);
        req.setAttribute("data",d);
        req.setAttribute("Consulta",cons);
        req.setAttribute("Incidencia",insi);
        req.setAttribute("Peticio",pet);
        req.setAttribute("InterI",interI);
        req.setAttribute("InterP",interP);
        req.setAttribute("tipus","graficac");
        return forward;
}
```

En aquesta acció podem veure els dos forwards que es defineixen: si l'acció té exit, es cridarà a la URL del Servlet de gràfics, i si en canvi, no en té, es tancarà la sessió. Aquest últim s'utilitza en cas que la variable d'usuari s'esborri per exhauriment del temporitzador, el qual considerem com a motiu de sortida del sistema, i en cas de que s'intenti realitzar qualsevol acció, aquesta ho detectarà i l'enviarà al formulari principal. Tot seguit, es prepara la petició HTTP mitjançant la classe HttpServlet. Després es demana a la base de dades les entitats que responguin a la condició de temps. En aquest cas, es passa el primer dia de mes, i el ControlImplementation demana totes les consultes entre el dia 1 del mes i l'últim (restant 1 al dia 1 del següent mes, gràcies a la classe Calendar que permet aquesta operació). Un cop hem recollit totes les entitats, es creen els atributs necessaris per que el Servlet sigui capaç de generar el gràfic. Un cop s'ha finalitzat l'encapsulació es retorna el forward i surt la petició.

Per la banda del Servlet de generació de gràfics, es va implementar el codi següent:

```
protected void processRequest(HttpServletRequest request,HttpServletResponse response)
{
	response.setContentType("image/jpeg");
	XYDataset xydataset = null;
	JFreeChart grafica;
	String entrada = request.getParameter("tipus");
	System.out.println("Valor de entrada : " + entrada);
	if (entrada.equals("graficmi"))
	{
		Incidencia[] insi = (Incidencia[])request.getAttribute("Incidencia");
		System.out.println("Longitud de insi : " + insi.length);
		try
		{
			xydataset = createDatasetMI(insi);
			grafica = createChartMI(xydataset);
			OutputStream salida = response.getOutputStream();
			ChartUtilities.writeChartAsJPEG(salida,grafica,800,600);
			salida.close();
		}
		catch(Exception e)
		{
			System.out.println(e.toString());
			System.out.println("Reventa en la generacio del grafic MI");
		}
	}
	else if (entrada.equals("graficmc"))
	{
		// ...
	}
}

protected void doGet(HttpServletRequest request,HttpServletResponse response)
{
	try
	{
		processRequest(request, response);
	}
	catch(Exception e)
	{
		System.out.println(e.toString());
		System.out.println(request.toString());
		System.out.println("Error en processar el GET");
	}
}

public void doPost( HttpServletRequest request,HttpServletResponse response )
{
	try
	{
		doGet(request, response);
	}
	catch(Exception e)
	{
		System.out.println(e.toString());
		System.out.println(request.toString());
		System.out.println("Error en processar el POST");
	}
}
```

Aquest codi s'executa en el Servlet, i s'encarrega de recollir la petició, convertir les dades i retornar el resultat. Tant si la petició arriba en forma de GET o bé de POST, el Servlet la recull en forma de classe doGet(HttpServletRequest request(petició), HttpServletResponse response(resposta). El gràfic es retorna al navegador en forma de imatge JPG, com es pot veure en la primera línia de la classe processRequest. Després es defineix dues variables, el dataset i la grafica pròpiament dita. El primer representa el conjunt de dades numèriques que definiran el gràfic, en concret, valors X i valors Y, el segon representa el conjunt de dades del gràfic. Tot seguit, s'empra el mètode createDatasetMI(Incidències insi), per convertir el conjunt d'incidències en un dataset, comptant les

coincidències per data i afegint-les a la sèrie temporal. A partir del dataset generat, creem la gràfica amb createChartMI(XYDataset dataset) i finalment fiquem en el canal de sortida el resultat JPG dels gràfics amb el mètode writeChartasJPEG. Per acabar, es finalitza amb el mètode close del OutputStream que el que fa és incorporar la gràfica a la resposta HTTP.

La solució adoptada per a la generació de informes dinàmics en format RTF implica l'accés a disc. El algorisme designat funciona agafant una plantilla del sistema de fitxers, i modificant-la amb els camps que s'adjunten a la entitat. L'accés a disc es fa seguint el camí marcat per la configuració enregistrada a la base de dades. Aquesta es configurable a través del menú Opcions de l'aplicatiu. Després es llegeix i s'introdueix en una variable String. Després es substitueixen les marques de la plantilla pels valors de l'entitat i s'adjunta en el correu electrònic. El codi emprat és el següent:

```
Correu M = new Correu();
M.setDestinatari(I.getCorreu());
M.setRemitent(correuDB.getNOMUSUARI());
M.setAssumpte("INCIDENCIA :"+ I.getCodi());
M.setCos(correuDB.getCOSINCIDENCIA());
M.setCC("");
M.setData(Calendar.getInstance().getTime());
M.setHost(correuDB.getSMTPHOST());
M.setIsHtml(true);
MimeBodyPart mbp = new MimeBodyPart();
String PathRTF = correuDB.getRUTARTF();
FileReader FR = new FileReader(PathRTF);
BufferedReader BR = new BufferedReader(FR);
String COSRTF = "";
String temp = BR.readLine();
while (temp != null)
{
        COSRTF = COSRTF + temp;
        temp = BR.readLine();
}
COSRTF = COSRTF.replaceAll("//usuari",I.getUsuariResponsable());
COSRTF = COSRTF.replaceAll("//codi",I.getCodi());
COSRTF = COSRTF.replaceAll("//data",I.getDataComunicacio().toString());
COSRTF = COSRTF.replaceAll("//hora",Calendar.getInstance().get(Calendar.HOUR) + ":" +
Calendar.getInstance().get(Calendar.MINUTE) );
COSRTF = COSRTF.replaceAll("//nom",I.getNom());
COSRTF = COSRTF.replaceAll("//cognoms",I.getCognoms());
COSRTF = COSRTF.replaceAll("//UA",I.getUnitatAdministrativa());
COSRTF = COSRTF.replaceAll("//telefon",I.getTelefon());
COSRTF = COSRTF.replaceAll("//estat",I.getEstat());
COSRTF = COSRTF.replaceAll("//tipologia",I.getTipologia());
COSRTF = COSRTF.replaceAll("//consulta",I.getIncidencia());
mbp.setDataHandler(new DataHandler(COSRTF,"text/html"));
mbp.setFileName("sol_as_0.rtf");
M.setRTF(mbp);
if(this.send(M))
        return true;
else
        return false;
// ...
```

Com podem veure, primer es genera el correu electrònic, emplenant les dades necessàries per a les capçaleres amb la configuració SMTP emmagatzemada a la base de dades i recuperada en temps d'execució. Observem que primer recuperem el camí del model amb el mètode getRUTARTF(), i tot seguit llegim el fitxer amb l'ajuda del FileReader BufferedReader. Després passem a substituir les marques de la plantilla (//usuari, //codi, ...) per els valors reals de la entitat.

# 8 PROCÉS DE VALIDACIÓ

## *8.1 Descripció del procés de desenvolupament*

Un cop finalitzat el desenvolupament de l'aplicatiu, procedirem a enumerar les fases en que s'ha trobat aquest projecte.

## *8.2 Proves en temps de desenvolupament*

Les proves en temps de desenvolupament anaven molt lligades a les tasques de depuració. Segons la metodologia de treball "Programació Extrema", aplicada a un únic treballador.

**Tests de funcionament amb el client :** El client de l'aplicatiu es concentra en una persona del Departament d'Informàtica, el rol de la qual és la de dirigir i supervisar totes les operacions dins del servei de Helpdesk, i forma part del òrgan directiu del propi Departament d'Informàtica. Per aquest motiu, s'estableix un estret lligam amb el client. Per contra, el rol que té la persona de contacte, no permet tenir la disponibilitat de temps desitjable per al desenvolupament conjunt de l'aplicatiu. A més, a això li em de sumar el fet de que tot el Departament es troba immers en un procés de trasllat d'emplaçament, a més, d'una migració de solucions per als serveis informàtics.

Per tant, la comunicació amb el client, es limita a proves concretes, prèvia reserva, per tal de que el servei es pugui dedicar a provar el software. Quan es van produir les visites, aquestes es basaven en la presentació del progrés de l'aplicatiu, una demostració de les funcionalitat implementades fins al moment, i una entrevista amb la intenció de recopilar el màxim de detall sobre el que el client espera de l'aplicatiu.

Aquestes visites tenien que anar preparades, amb una presentació i un qüestionari, així s'aconsegueix una recopilació ordenada de les dades que es necessiten per implementar l'aplicatiu, i se li ofereix al client una explicació prèvia de què és el que veurà quan proba el programa.

La primera acció, un cop definit el treball a fer, va ser contactar amb el client per determinar les necessitats i requeriments amb més extensió. D'aquesta manera es podia donar forma al programa, ja que tot i que els requeriments són clars, fa falta una visió global de l'usuari, per matisar el programa.

Per desenvolupar l'aplicatiu, són moltes les possibilitats de configuració de l'entorn d'usuari, tant de manera lògica com gràfica. Per a definir els objectius del client en aquest sentit, la visita al client i una entrevista amb preguntes concretes, permet obtenir les respostes que es plantegen alhora de donar forma a un aplicatiu.

En aquesta visita, el que es va fer va ser comentar la intenció del programador, descrivint una estructura proposada pel programador, amb la particularitat de deixar que el client reaccioni i respongui a les proposicions del programador. Per a la majoria de clients resulta més difícil expressar amb paraules els requeriments del seu futur entorn. Proposar un model representa una ajuda alhora de configurar un model a mesura del client.

La visita es va completar amb un diagrama de lògica de l'aplicatiu, a imatge d'un mapa. De manera visual el client podia identificar els passos a realitzar per completar les accions que desitja realitzar amb l'aplicatiu. Es necessari brindar d'alguna forma la possibilitat de que el client mostri les seves opinions, i d'acord amb el programador, s'acordi una estructura que resulti del compromís entre comoditat per a l'usuari i facilitat de programació, ja que en la realització de projectes de aplicatius el

factor temps es un paràmetre molt important, de manera que el programador buscarà solucions que permetin codificar l'aplicatiu en el menor temps possible.

Les proves podem dir que es separa en dues fases : la fase de producció i la fase de refinament.

Anomenarem fase de producció a les proves realitzades a l'estació de treball de desenvolupament. Aquí s'engloben les pròpies proves de funcionalitat, aspecte, rendiment, etc. Cada modificació que es realitza sobre l'aplicació Web es pot comprovar immediatament mitjançant l'actualització de l'aplicació carregada al Weblogic. Mitjançant un navegador Web, en aquest cas Mozilla Firefox ( versió actual : 1.5.9 )

El que hem anomenat fase de refinament engloba les proves realitzades amb el client. Això només es va donar un cop, però sobre una versió bastant depurada de l'aplicació Web final.

## *8.3 Relació amb el client*

Les relacions amb el client podem definir-les de molt concretes. En primer lloc, ens van fer arribar una petició amb les característiques desitjades. A partir d'aquesta petició s'han donat lloc dues reunions, amb una tercera concertada.

**Primera visita:** En la primera es va produir durant una fase prematura del programa, i estava destinada a familiaritzar-se amb el client, l'entorn de treball, etc.

Abans de començar a desenvolupar el disseny inicial, es va fer una posada d'acord amb el client, sobre les funcionalitats requerides i la contrastació de les seves peticions en front a la redacció dels requeriments de l'aplicatiu.

La primera part de la visita, estava destinada a concretar els hàbits de ús del client. Preguntes com sobre quina plataforma s'executarà l'aplicatiu, la càrrega de treball, el nombre d'usuaris que romandran actius en l'aplicatiu, servien per donar una visió àmplia de com es tenia que desenvolupar l'aplicatiu, i quines serien les prioritats de disseny de l'aplicatiu.

La segona part estava destinada a donar forma a la informació que es va proporcionar com a requeriment. Això es, analitzar amb el client, els camps de les taules de la bases de dades. Per a tal, es van definir les classes, i s'anava preguntant al client les propietats que aquestes havien de tenir.

La tercera part, era la de demanar un canal de comunicació, això és, demanar horaris de visites, etc.

**Segona visita:** En la segona es va ser amb un aplicatiu funcional a un 20 o 30 %. Integrava ja la autenticació i la adició, consulta i eliminació de consultes. No incorporava estils de pàgina, ni les funcionalitats afegides. Incorporava la funcionalitat bàsica, i proporcionava una primera visió de la lògica de l'aplicatiu : formularis, pàgines web, llistats, etc.

Els objectius d'aquesta visita eren :

- Confirmar el modelat de dades
- Provar la lògica de navegació de l'aplicatiu
- Confirmar les accions necessàries per a la gestió de les incidències

En la recepció de peticions sobre l'aplicatiu es van remarcar les següents conclusions :

- Reforma de les funcionalitats, i remodelat de l'estructura de dades
- Inclusió del auto - tancament
- Remodelació dels formularis

**Tercera visita :** La última visita, la ja definitiva es va fer sobre la prova. Amb l'ajut de les operadores es va fer un test. Amb un portàtil integrant el servidor Weblogic i Oracle connectat a la xarxa del client, aquestes podien accedir a l'aplicatiu. Aquesta visita suposava el primer contacte de les operadores i l'aplicatiu. Per tant, suposava el primer recull d'opinions sobre l'entorn gràfic, la distribució dels elements, etc.

Objectius :

- Analitzar l'entorn gràfic de l'usuari.
- Validar la funcionalitat, eficiència i velocitat del servidor.
- Comprovar el funcionament de la secció "Informes".
- Comprovar el funcionament multi - usuari i simultani.

Conclusions :

- Implementació de correus electrònics amb informe RTF auto - generat.
- Remodelació dels formularis.
- Inclusió d'un nou sistema de codificació per a la identificació de les incidències.

**Quarta visita :** La última visita, va correspondre amb la prova final, i l'entrega de l'aplicatiu. Es va incorporar l'aplicatiu a un servidor de la xarxa informàtica del client, i es va procedir a la introducció i gestió de casos reals. Aquesta prova es va dur a terme durant un període de dos hores, atenent als usuàries per via telefònica i per e-mail. La secció de recepció de e-mails no es va utilitzar per el motiu de unificar els serveis e-mails en una sola aplicació (Microsoft Outlook). Els objectius d'aquesta prova :

Comprovar el funcionament en casos reals.
Comprovar les últimes modificacions realitzades.

A les dues hores es van introduir els usuaris reals, comprovat el funcionament de les variables (com ara unitats administratives, etc.), introduïdes consultes i incidències, intervencions, i comprovats els resultats al final del període mitjançant l'opció d'Informes.

## *8.4 Anàlisis del procés de validació*

Per a la validació prèvia a l'entrega definitiva de l'aplicatiu, es va procedir a un test per part del client, en una màquina no definitiva, però que acomplia els requisits bàsics. El test proposat pel client consistia en un procés de introducció i gestió de dades dintre dels cassos estimats que esdevenen en el servei de Helpdesk. Es va establir un periòdic de fase Beta, durant el qual el client havia d'informar dels errors o desajusts en el funcionament, d'acord amb les especificacions requerides amb anterioritat. Aquest període es va exhaurir, sense rebre cap notificació del client, el que va dur a pensar que no s'havia produït la validació del aplicatiu. En cas de rebre un informe per part del client, s'hagués estudiat l'origen del mal funcionament i s'hagués arreglat en la còpia de l'aplicatiu del client, i s'hagués afegit de l'errada i la seva solució en el Manual del Programador, que es pot trobar en l'annex d'aquest treball. Per tant, l'anàlisi sobre aquest procés de validació no es pot efectuar sense considerar que aquest procés no s'ha efectuat degudament.

# 9 CONCLUSIONS I OBSERVACIONS

Al finalitzar el desenvolupament d'aquesta aplicació, es procedeix a l'anàlisi de la pròpia aplicació, i paral·lelament s'analitza el procés teòric i pràctic de desenvolupament de l'aplicació:

**Problemes.** L'aplicació es va desenvolupar partint d'una base en que els coneixements eren baixos i per tant, el desenvolupament es basava en la implementació seqüencial de les funcionalitats. En les proves es podien detectar errors de codificació o errors de execució. Mitjançant l'entorn de desenvolupament, els errors de codificació poden ser detectats en gran majoria. Per altra banda, els errors en temps d'execució es detectaven a través de diversos missatges.

Alguns errors d'execució es detectaven mitjançant respostes HTTP amb codi 500. Altres errors es mostren mitjançant el control d'errors de Apache Struts. Alguns retornen una pàgina web, on es retornen detalladament els errors que s'han produït.

Respecte a la previsió inicial, el desenvolupament final es va desviar molt en el temps previst, la data d'entrega es va anar allargant, en gran mesura degut a un conjunt de factors :

El primer i més important va ser la falta d'experiència tant l'entorn de desenvolupament com d'execució, la manca de documentació que permeti a un programador entrar ràpidament a programar, degut a l'abstracció del entorn, i a la inclusió de un gran nombre de capes entre l'execució i la presentació.

El segon factor va ser degut  a un error de la versió de desenvolupament de BEA Weblogic en que les compilacions de l'aplicatiu, per realitzar les proves de codificació, s'endarrerien degut a un bug en el servidor Weblogic, aquest defecte no es va solucionar fins al cap de 8 mesos amb el Maintance Pack 1.

El tercer factor va ser que degut al desenvolupament dintre de l'empresa ADTEL, la gestió del projecte va ser completa, i per tant, van esdevenir aspectes que no afecten en altres entorns de treball, com ara la coincidència de dates amb la migració del sistema informàtic i el període de vacances, el que va provocar ambdues parades en el desenvolupament, la dedicació en altres projectes de l'empresa, el retràs per dates de visita, etc.

En relació a la implementació en sí, aquests retards van motivar que :

L'especificació de l'aplicatiu va patir unes demores, ja en temps de desenvolupament, perquè el client va anar demanant noves funcionalitats, com ara el sistema de missatgeria, però en global no es va dilatar en excés.

El disseny de les classes en aquest cas tampoc es va dilatar en excés ja que d'un primer moment partíem de la informació que es podia demanar a l'usuari. L'única variació va ser deguda al format de llistats desitjats pel client, que va obligar a modificar algunes classes per fer-les més intel·ligibles al moment de mostrar-les per pantalla.

El disseny dels mètodes va ser uns dels més perjudicats, ja que aquests van estar sent remodelats contínuament, a mesura s'anaven descobrint les possibilitats de cadascuna de les funcionalitats que ofereix BEA Weblogic.

**Conclusions**. Les conclusions que es poden extreure del desenvolupament d'una aplicació Web d'aquestes dimensions són diverses.

La primera d'elles es refereix a la plataforma d'aquesta solució creada. S'ha desenvolupat una aplicació sobre una plataforma d'una capacitat molt superior a la necessitada per l'aplicatiu, tot i els avantatges que a la vegada proporciona a l'hora d'oferir eines per a un desplegament ràpid. Aquesta plataforma està preparada per treballar en condicions de càrrega elevada, amb servidors 24x7. Ofereix una gran quantitat de possibilitats amb tot el que sigui referent a lògica empresarial, entorns corporatius, etc. Destaquen la gestió de diversos servidors per al servei d'una aplicació Web, la incorporació d'una màquina virtual de Java pròpia, optimitzada per aquestes funcions i la integració estreta que s'ha aconseguit entre servidor i entorn de desenvolupament, que en aquest cas es tracta d'un de gran qualitat, Eclipse. Amb això, el fet de desenvolupar una petita aplicació Web, en que no hi han requeriments crítics, es fa difícil en el sentit de que pot recarregar el codi amb funcionalitats afegides, o bé, es té que optimitzar l'aplicatiu per que tingui un funcionament eficaç sobre una plataforma tan pesada. Tot i això, aquesta plataforma es mostra molt eficaç un cop s'ha desplegat l'aplicatiu, i ja s'han efectuat algun cop les accions pertinents. Quan s'inicia l'aplicatiu i es van fent accions per primer cop, aquestes s'han d'interpretar i després executar, el que fa que l'experiència del usuari no sigui en un primer moment, satisfactòria. Un cop el servidor porta un cert temps d'utilització, aquest va mostrant les seves virtuts, no es degrada, i ofereix un temps de resposta molt bo. No mostra errors motivats per la inestabilitat del sistema, i sempre està disponible.

En la referència al que fa l'aplicatiu, l'anàlisi del procés de desenvolupament se'n poden extreure diverses lectures per poder optimitzar el temps i millorar els resultats.

**Punts forts i punts febles de la preparació per a la execució d'un projecte de desenvolupament de software.** Des del punt de vista de l'estudiant, l'anàlisi del projecte serveix per destacar els punts forts i febles de la formació de la fase acadèmica. El treball es separa en tres grans parts : organització, programació i documentació.

La organització de les tasques ha esdevingut molt personal, degut a que no és un projecte en el que hagi participat un equip de programadors. En aquest aspecte, s'ha estructurat en la realització i assoliment de les següents fases:

1. Aprenentatge de l'entorn de treball.
2. Elaboració de una beta de l'aplicatiu, com a prototip.
3. Test amb el client.
4. Realització de l'aplicatiu definitiu.
5. Realització de les proves pertinents, i de la documentació associada.

Per al tancament de les fases anteriors, es va elaborar un calendari, amb unes previsions que es van esdevenir molt optimistes, degut a les dificultats trobades en cadascuna de les fases, a més d'elaboració d'altres tasques pròpies de l'empresa, que han ajudat a integrar l'estudiant en l'àmbit del treball.

Les fases que han esdevingut més difícils a títol personal s'han rellevat com la realització de l'aplicatiu definitiu, i la realització de les proves pertinents.

El fet de realitzar un aplicatiu comercial, implica que les funcionalitats s'han de complir sota qualsevol cas de dades entrades. Altres detalls, són la normalització dels formularis en favor de la seva facilitat d'ús, una estabilitat front problemes de connexió entre elements de l'aplicatiu i una gestió d'errors eficaç i amigable per a l'usuari.

L'altra dificultat important en aquest projecte també ha esdenvingut el aprenentatge de l'entorn de treball. Partint d'una base de coneixement de la tecnologia de programació Java i els Servlets, l'entorn BEA Weblogic, la seva programació i l'administració del servidor, unit a l'administració i programació de la base de dades Sun Oracle 8i, representen un gran salt en l'àmbit de la programació.

El punt més fort de la preparació acadèmica és la programació. Amb la base teòrica de treball amb el llenguatge Java i les seves característiques, es crea l'aplicatiu amb una velocitat i qualitat de programació adequades. Els reptes més importants de la programació són l'aprenentatge de l'entorn del servidor d'aplicacions BEA Weblogic, degut a la integració dintre seu de diverses tecnologies que traduïen les capes entre l'aplicació i l'usuari, com podria ser la tecnologia Apache Beehive, que es basa en etiquetes pròpies dintre la tecnologia JSP de pàgines Web, canviant la dinàmica de les mateixes i proporcionant una versió més potent, però més abstracta de la pròpia pàgina. Per altra banda, destacaríem la manca de problemes greus a l'hora de desenvolupar algorismes per dotar a l'aplicatiu de les funcionalitats desitjades, traient de banda l'apartat de generació de gràfics representatius d'informes, cas en el que s'ha recorregut a una llibreria externa, alliberant d'una carga superior en el cas de tenir que implementar cadascun dels elements gràfics que conformen un gràfic d'aquestes característiques.

El punt més fluix per part de la formació adquirida ha esdevingut la direcció del projecte en la seva totalitat a més d'aprendre nous coneixements tècnics en temes de programació de aplicacions Web, noves tecnologies i bases de dades.

**Experiències adquirides**

Experiència en un projecte de desenvolupament de software complet. A més, sobre una plataforma molt desenvolupada, a diferència de altres projectes que s'han desenvolupat en la vida acadèmica, sobre servidor molt simples, sense explorar les possibilitats i també les varietats de servidors i plataformes, i amb una gestió del projecte molt més senzilla (interacció amb el client, temps de desenvolupaments, creació de la documentació, etc.)

La segona experiència és la importància d'una programació del temps destinat a desenvolupar cada tasca. Tenint en compte que per a fer cada tasca, s'havia de perdre molt de temps consultant informació, i tot i així, es perdia temps en implementacions equivocades, ressalta la idea de tenir organitzats els passos de l'aplicatiu, tot i dependre d'altres factors per poder començar aquest desenvolupament (documentació, directrius, etc.). Amb una gestió eficaç del temps, l'aplicatiu podria haver experimentat millores clares de funcionalitat.

La tercera experiència es relació amb el client. És evident que el client no ha de saber quina informació li requerim, ni tampoc ha de poder oferir-nos el temps que seria necessari per fixar amb claredat els requeriments de l'aplicatiu, però hem d'intentar arribar a aquest nivell, ja que el resultat d'una col·laboració estreta entre el programador i el client ofereix un aplicatiu finalitzat en menys temps i amb més especialització per al client.

**Possibles millores**

Entre les millores que es detecten que se li poden implementar a l'aplicatiu, són les següents:

- Optimització de la gestió d'errors. Implementar un control d'errors que permeti no perdre informació i/o temps entre processos permet elevar el nivell de qualitat de l'aplicatiu. Per exemple, a través d'un sistema d'autorecuperació a partir de plantilles, que permeti restaurar informació a partir de plantilles d'informació.

- Optimització de les consultes SQL. Per exemple, es podrien millorar les consultes permetent realitzar cerques en múltiples criteris de manera ràpida, utilitzant més eficientment les consultes SQL.

- Implementació d'un registre d'activitat. Per augmentar la seguretat del funcionament, es podria afegir un sistema de registre d'activitat dels usuaris. Això permetria, en cas de fallades, traçar el comportament de l'aplicatiu.

# 10 ANNEXOS

## *10.1 Glossari*

**Java :** Llenguatge de programació de alt nivell, basat en la Programació orientada a Objectes, mitjançant les classes.

**Classe :** Estructura lògica que conté mètodes i propietats.

**Mètode :** Són parts de codi que implementen una funcionalitat amb l'ajut de les propietats i els paràmetres.

**Propietat :** Són valors, que poden ser de qualsevol forma, que estan confinats dintre la classe a la qual pertanyen.

**Paràmetres :** Són valors que es transmeten als mètodes per a realitzar funcions.

**BLOB** : Grans objectes binaris, *Binary large object*. Tipus de variable que permet emmagatzemar informació a nivell de bit, com per exemple, imatges.

**CLOB** : Grans objectes de caràcters, *Character large object.* Tipus de variable que permet emmagatzemar informació a nivell de caràcter, com per exemple, text cru, codi font.

***Redo*** : Esdeveniment que fa que les sentències SQL que s'han efectuat prèviament a la memòria, s'efectuïn finalment al disc.

***Rollback*** : Instrucció que permet emmagatzemar en una cache les sentencies fetes. D'aquesta manera, s'estableixen punts de retorn .

**JDBC** : Java Data Base Connectivity. API que permet la interconnexió entre aplicacions Java i gestors de bases de dades DBMS.

**ODBC** : Open Data Base Connectivity. Estàndard de accés a gestors de bases de dades desenvolupat per Microsoft.

**SSH** : Secure Shell. Consola segura, protocol de comunicació per a terminals en que la comunicació es considera segura per la encriptació emprada.

**JSP** : Java Server Pages. Pàgines HTML que incorporen codi Java. Aquest codi és interpretat pel servidor abans de que s'enviïn contra el client HTML, executant mètodes i substituint propietats pel seu valor real. Equival a dir que són una forma de pàgines HTML dinàmiques.

**SQL** : Simple Query Language. Llenguatge utilitzat per a la comunicació entre aplicacions i bases de dades. Es basa en un llenguatge de alt nivell que permet introduir dades, fer consultes, modificar taules i esborrar registres.

**Enter. *Integer*** : Tipus de variable numèrica, limitada en valor, no admet fraccions.

**Cadena. *String*** : Conjunt de caràcters alfanumèrics.

**SMTP** : Simple Mail Transport Protocol. Protocol d'enviament de correu utilitzat en Internet.

**RTF** : Rich Text Format. Format de text enriquit, desenvolupat per Microsoft, està basat en un sistema d'etiquetes similar a les de XML.

**API** : Application Programming Interface. Interfície per a la programació de aplicacions. Proporciona mètodes i propietats al programador a l'hora de treballar amb diferents entitats, per exemple, bases de dades, dispositius d'entrada/sortida, etc.

**XML** : eXtensive Markup Language. Llenguatge de marques extensible. És un llenguatge d'alt nivell, basat en etiquetes que confinen les dades. En la definició de XML, les etiquetes defineixen el tipus d'informació, i el cos entre etiquetes, conformen les dades. A més, aquestes etiquetes s'utilitzen de forma jeràrquica, de manera que poden crear diferents pseudo-llenguatges a partir de XML.

**AJAX** : Asynchronous Javascript  Xml. Mètode que permet intercanviar informació al servidor amb el client sobre HTTP, sense haver de recarregar la pàgina. Utilitza un objecte de Javascript en que es comunica amb el servidor, i aquest li retorna dades dinàmiques, les quals es poden executar en el client durant el temps d'execució.

**HTTP** : Hyper Text Transport Protocol. Protocol de transport d'hipertext. Protocol primari sobre el qual treballen les pàgines Web. Permet la petició de pàgines i l'enviament de dades cap al servidor.

**DataSet** : Objecte que recull dades en funció d'un altres camp, per exemple, un XYDataset gestiona dades en funció d'unes ordenades (X)

## 10.2 Referències.


**BEA Weblogic**
http://www.bea.com
http://www.dev2dev.bea.com
http://edocs.beasys.com/workshop/docs81/doc/en/core/index.html
http://edocs.bea.com/wls/docs81/adminguide/overview_domain.html

**SUN Oracle**
http://www.rocket99.com/oracle/oracle5.html
http://www.techonthenet.com/index.php

**Llenguatge SQL**
http://www.techonthenet.com/index.php

**APACHE Tomcat**
http://wiki.apache.org/tomcat/

**APACHE Struts**
http://wiki.apache.org/struts/FrontPage

**APACHE Beehive**
http://beehive.apache.org/docs/1.0.2/

**J2EE**
http://java.sun.com/j2ee/sdk_1.3/techdocs/api/overview-summary.html

**JFreeChart**
http://www.iana.org/assignments/media-types/

**Llistat de Servlets Java per realitzar gràfiques open-source**
http://www.java-source.net/open-source/charting-and-reporting

**Java Mail**
http://java.sun.com/products/javamail/javadocs/index.html
http://www.programacion.net/java/tutorial/javamail/

## *10.3 Bateria de proves*

Per detectar els errors, s'utilitzaven proves típiques de funcionalitat. Aquestes, en conjunt, les podem enumerar en el següent llistat

**Autenticació:**
Autenticació d'un usuari no existent.
Autenticació d'un usuari de forma errònia : en aquest cas, password incorrecte.
Autenticació d'un usuari amb permisos limitats, amb l'objectiu de comprovar les limitacions aplicades.
Autenticació d'un usuari amb plens permisos, amb l'objectiu de comprovar el lliure accés a les funcions de l'aplicació.

**Consultes:**
Creació d'una consulta de forma errònia : Format erroni d'un camp.
Creació d'una consulta correcta.
Cerca d'una consulta que no existeix. Test de cadascun dels mètodes de cerca.
Cerca d'una consulta única. Test del resultat.
Cerca d'una consulta múltiple. Test del resultat.
Cerca d'una consulta esborrada. Test del resultat.
Cerca d'una consulta que està tancada. Test del resultat.
Modificació d'una consulta de forma errònia amb el format erroni en un camp.
Modificació correcta d'una consulta.
Intent d'esborrar una consulta. No es confirma la eliminació.
Intent d'esborrar una consulta. Es confirma l'eliminació.

**Incidències:**
Creació d'una incidència de forma errònia, tipus 2 : Format erroni d'un camp
Creació d'una incidència correcta.
Cerca d'una incidència que no existeix. Test de cadascun dels mètodes de cerca.
Cerca d'una incidència única. Test del resultat.
Cerca d'una incidència múltiple. Test del resultat.
Cerca d'una incidència esborrada. Test del resultat.
Cerca d'una incidència que està tancada. Test del resultat.
Modificació d'una incidència de forma errònia: Format erroni d'un camp
Modificació correcta d'una incidència
Intent d'esborrar una incidència. No es confirma la eliminació.
Intent d'esborrar una incidència. Es confirma l'eliminació.

**Peticions:**
Creació d'una incidència de forma errònia, tipus 1 : Manca un camp.
Creació d'una incidència de forma errònia, tipus 2 : Format erroni d'un camp.
Creació d'una incidència correcta.
Cerca d'una incidència que no existeix. Test de cadascun dels mètodes de cerca.
Cerca d'una incidència única. Test del resultat.
Cerca d'una incidència múltiple. Test del resultat.
Cerca d'una incidència esborrada. Test del resultat.
Cerca d'una incidència que està tancada. Test del resultat.
Modificació d'una incidència de forma errònia, tipus 1 : Manca un camp.
Modificació d'una incidència de forma errònia, tipus 2 : Format erroni d'un camp.
Modificació correcta d'una incidència.
Intent d'esborrar una incidència. No es confirma la eliminació.
Intent d'esborrar una incidència. Es confirma l'eliminació.

**Intervencions:**
Adició correcta d'una intervenció a una incidència.
Adició errònia d'una intervenció a una incidència.
Adició correcta d'una intervenció a una petició.
Adició errònia d'una intervenció a una petició.
Modificació d'una intervenció (incidència).
Modificació d'una intervenció (incidència) amb errors.
Modificació d'una intervenció (petició).
Modificació d'una intervenció (petició) amb errors.
Eliminació d'una intervenció (incidència).
Eliminació d'una intervenció (petició).

**Informes:**
Petició d'un informe anual complet.
Petició d'un gràfic corresponent a un informe anual complet.
Petició d'un informe mensual complet, quan no hi ha cap element registrat.
Petició d'un gràfic a partir del informe mensual complet anterior.
Petició d'un informe mensual complet, quan hi ha una part amb elements registrats.
Petició d'un gràfic generat a partir del informe mensual complet anterior.
Petició d'un informe mensual complet, quan totes les incidències han registrat elements.
Petició d'un gràfic fet del informe anterior.
Petició d'un informe mensual d'incidències, quan no hi ha incidències enregistrades.
Petició d'un gràfic a partir d'un informe mensual d'incidències com l'anterior.
Petició de realització d'un informe mensual d'incidències quan hi han incidències emmagatzemades.
Petició d'un gràfic corresponent a un informe mensual d'incidències com l'exemple anterior.

**Usuaris:**
Creació d'un usuari, amb les dades coincidint amb el format esperat.
Creació d'un usuari, sense complir amb el format.
Recuperació per a la modificació d'un usuari.
Modificació d'un usuari.
Modificació errònia d'un usuari.
Eliminació d'un usuari.

**Àrees:**
Creació d'una nova àrea, amb el format conforme al esperat.
Creació d'una nova àrea, amb el format erroni.
Eliminació d'una àrea.

**Estats:**
Creació d'un nou estat, d'acord amb el format establert.
Creació d'un nou estat, de forma errònia.
Eliminació d'un estat.

**Correu:**
Enviament d'un correu, amb èxit.
Enviament d'un correu, sens èxit.
Recepció dels correus.

**Tancar Any:**
Introducció d'un any de tancament posterior al prèviament enregistrat.
Introducció d'un any de tancament anterior al prèviament enregistrat.

**Unitat Administrativa:**
Creació d'una nova Unitat Administrativa, amb el format correcte.
Creació d'una nova Unitat Administrativa, provocant un error.
Eliminació d'una Unitat Administrativa.

**Responsable de UA :**
Adició d'un responsable de Unitat Administrativa, sense errors.
Adició d'un responsable de Unitat Administrativa, amb errors.
Eliminació d'un responsable de Unitat Administrativa.

**Tipologies:**
Adició d'una nova tipologia.
Adició d'una nova tipologia, expressament errònia.
Eliminació d'una tipologia existent.

**Responsables d'àrea:**
Creació d'un nou responsable d'àrea.
Creació d'un nou responsable d'àrea, expressament errònia.
Eliminació d'un responsable existent.

# *11 Continguts utilitzats en la relació amb el client*

## 11.1 Primera visita

**ÍNDEX**

## 1.- INTRODUCCIÓ I DESCRIPCIÓ DE LA LÒGICA

El software és basa en un servei Web, que consta d'un servidor, que conté l'aplicació i de diversos clients, que són en aquest cas, clients Web com ara Internet Explorer i el Mozilla Firefox. El servidor executa una aplicació que es basa en un software format pel programa BEA Weblogic Server, que és el servidor de l'aplicació, i el servidor de bases de dades anomenat Oracle. Junts ofereixen la funcionalitat de l'aplicació Helpdesk.

Per a utilitzar-lo, els clients Web han d'introduir una adreça Web que ens portarà a l'aplicació. Un cop hem accedit a l'aplicació, el primer que ens apareix és el formulari de Autenticació de Usuaris o "Login". Aquí els usuaris han d'introduir un nom i una paraula clau, el que permet que només les persones autoritzades accedeixen a l'aplicació.
Un cop hem accedit, el software ens dirigeix a una altra plana web que ens permet seleccionar la funció a desenvolupar.

En primera instància, les opcions possibles són :

- Consultes
- Incidències
- Usuaris
- Altres

### 1.1.- Consultes

Aquesta secció, seguint la proposta original, va destinada a les assistències que son resoltes telefònicament. No impliquen a la resta de departaments del Servei tècnic.
Les opcions que en primer terme se'ns ofereixen seran :

- Buscar per Estat
- Buscar per Data
- Buscar per Usuari
- Buscar per Numero de Consulta
- Nova Consulta

Òbviament, les quatre primeres ens permeten accedir a consultes ja emmagatzemades, oferint diverses vies per accedir-hi. Un cop hem recuperat la consulta buscada, podem realitzar les següents accions:

- Modificar
- Esborrar
- Afegir Actuació

Les dues primeres són necessàries per a la gestió de les consultes, i la última, ens permet guardar les solucions proposades i les dades relatives a aquestes solucions. En tot cas, les solucions definitives també són enregistrades a la Consulta, de manera que separem l'històric d'actuacions en un cas concret, de les solucions exactes als problemes determinats.

1.2.- Incidències

Les Incidències fan referència a aquells contratemps que exigeixen la intervenció de algun dels departaments del Servei Tècnic. Per exemple, el de Hardware. Aquest cas no exclou la possibilitat que el problema no pugui ser resolt de manera telefònica. Però s'utilitza aquesta diferenciació per agrupar les intervencions externes. En una primera visió es va separar les incidències en primer i segon nivell, segons l'assistent d'incidència, però de moment, s'ha obviat aquesta classificació per simplificar els conceptes.

El sistema d'enregistrament de dades és molt similar al de Consultes, però incorpora la funcionalitat de intercanviar missatges entre els diferents departaments.

Quan s'obre una incidència, i aquesta s'assigna a una secció, aquesta rep un correu en les dades de la incidència. Així és redueix el temps d'intervenció i es delega l'assistència a l'usuari.

De manera anàloga, s'ha previst un registre d'actuacions en previsió que la incidència causi diverses actuacions

### 1.3.- Altres

En aquesta secció hem englobat els serveis i les configuracions auxiliars. Aquestes, en una segona instància, proposem que siguin les següents :

- Gestió d'usuaris
- Bloc de Notes
- Control dels Estats
- Control de les Àrees i Responsables adjunts

La Gestió dels usuaris ens permet entre altres coses, guardar les dades dels operaris Helpdesk, i configurar les seves paraules claus, entre altres opcions. El Bloc de Notes esdevé un servei auxiliar on anotar dades de manera no classificada, per a determinats casos, a voluntat del operari. El control d'estats i de les àrees ens permet variar en qualsevol moment la classificació dels estats i de les àrees, en avançament a variacions de la jerarquia organitzativa.

## 2 TAULES DE DADES

En aquesta secció oferim una relació de la distribució de les dades i els seus camps, per comentar i ajustar la proposta a les necessitats del servei Helpdesk.

### Taula Actuacions - Consulta

| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| IDACTUACIO | NUMBER | 0 | No | | INDEX |
| DATAACTUACIO | DATE | 7 | No | | |
| DESCRIPCIOACTUACIO | VARCHAR2 | 10 | Yes | | |
| USUARIRESPONSABLE | NUMBER | 0 | No | | TECNIC ACTUACIO |
| NUMCONSULTA | NUMBER | 0 | No | | REFERENCIA |

### Taula Actuacions – Incidència

| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| IDACTUACIO | NUMBER | 0 | No | | INDEX |
| DATAACTUACIO | DATE | 7 | No | | |
| DESCRIPCIOACTUACIO | VARCHAR2 | 10 | Yes | | |
| USUARIRESPONSABLE | NUMBER | 0 | No | | |
| USUARIAREA | NUMBER | 0 | No | | TECNIC ACTUACIO |
| NUMINCIDENCIA | VARCHAR2 | 10 | No | | REFERENCIA |

### Taula Àrees

| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| IDAREA | VARCHAR2 | 10 | No | | INDEX |
| NOMAREA | VARCHAR2 | 10 | No | | |
| SITUACIO | VARCHAR2 | 10 | No | | LOCALITZACIO |
| TELEFON | VARCHAR2 | 10 | No | | |
| CORREU | VARCHAR2 | 10 | No | | |
| DIRECTOR | VARCHAR2 | 10 | No | | |

**Taula Consulta**

| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| IDCONSULTA | NUMBER | 6 | No | | |
| NOM | VARCHAR2 | 40 | No | | |
| COGNOMS | VARCHAR2 | 60 | Yes | | |
| CORREU | VARCHAR2 | 100 | Yes | | |
| TELEFON | VARCHAR2 | 15 | Yes | | |
| DATACOMUNICACIO | DATE | 7 | No | | |
| TIPOLOGIACONSULTA | NUMBER | 0 | No | | CLASSIFICACIO |
| CONSULTA | VARCHAR2 | 1000 | Yes | | |
| SOLUCIOFINAL | VARCHAR2 | 1000 | Yes | | |
| ESTAT | VARCHAR2 | 3 | No | | |
| DATACONFORME | DATE | 7 | Yes | | |
| NUMINCIDENCIA | NUMBER | 6 | Yes | | SI S ESCAU |
| USUARIRESPONSABLE | VARCHAR2 | 10 | Yes | | TECNIC ASSIGNAT |

**Taula Estats**

| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| IDESTAT | VARCHAR2 | 10 | No | | |
| DESCRIPCIO | VARCHAR2 | 10 | No | | |

**Taula Incidència**

| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| NUMINCIDENCIA | NUMBER | 6 | No | | |
| NOM | VARCHAR2 | 40 | No | | |
| COGNOMS | VARCHAR2 | 60 | Yes | | |
| CORREU | VARCHAR2 | 100 | Yes | | |
| TELEFON | VARCHAR2 | 15 | Yes | | |
| DATACOMUNICACIO | DATE | 7 | No | | |
| TIPOLOGIAINCIDENCIA | NUMBER | 0 | No | | |
| INCIDENCIA | VARCHAR2 | 1000 | Yes | | |
| SOLUCIOFINAL | VARCHAR2 | 1000 | Yes | | |
| ESTAT | VARCHAR2 | 3 | No | | |
| DATACONFORME | DATE | 7 | Yes | | |
| USUARIRESPONSABLE | NUMBER | 0 | No | | |
| AREA | VARCHAR2 | 10 | No | | |

**Taula Intervencions - Incidència**

| Nom del camp | Tipus de dades | Long. | Nul·ls ? | Valor Inicial | Descr. |
|---|---|---|---|---|---|
| IDINTERVENCIO | ROWID | 10 | No | | |
| NOM | VARCHAR2 | 40 | No | | |
| COGNOMS | VARCHAR2 | 70 | No | | |
| TELEFON | VARCHAR2 | 15 | Yes | | |
| CORREUUSUARI | VARCHAR2 | 100 | Yes | | |
| DATAINICI | DATE | 7 | No | | |
| TIPUSINTERVENCIO | NUMBER | 1 | No | | |
| ESTAT | VARCHAR2 | 1 | No | | |
| AREA | VARCHAR2 | 10 | Yes | | |
| PERSONAAREA | VARCHAR2 | 10 | Yes | | |
| DESCRIPCIO | VARCHAR2 | 1000 | No | | |
| SOLUCIO | VARCHAR2 | 1000 | Yes | | |
| DATASOLUCIO | DATE | 7 | Yes | | |
| DATACONFORME | DATE | 7 | Yes | | |
| USUARICONFORME | NUMBER | 1 | Yes | | APROBACIO |
| USUARIRESPONSABLE | ROWID | 10 | No | | |
| INTERVENCIOANTERIOR | NUMBER | 6 | Yes | | REFERENCIA |
| NUMINTERVENCIONS | NUMBER | 0 | No | | COMPTADOR |

**Taula Intervencions – Consultes**

| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| IDINTERVENCIO | NUMBER | 10 | No | | |
| NOM | VARCHAR2 | 40 | No | | |
| COGNOMS | VARCHAR2 | 70 | No | | |
| TELEFON | VARCHAR2 | 15 | Yes | | |
| NUMINTERVENCIO | NUMBER | 6 | No | | |
| CORREUUSUARI | VARCHAR2 | 100 | Yes | | |
| DATAINICI | DATE | 7 | No | | |
| TIPUSINTERVENCIO | NUMBER | 1 | No | | |
| ESTAT | VARCHAR2 | 1 | No | | |
| AREA | VARCHAR2 | 10 | Yes | | |
| PERSONAAREA | VARCHAR2 | 10 | Yes | | |
| DESCRIPCIO | VARCHAR2 | 1000 | No | | |
| SOLUCIO | VARCHAR2 | 1000 | Yes | | |
| DATASOLUCIO | DATE | 7 | Yes | | |
| DATACONFORME | DATE | 7 | Yes | | |
| USUARICONFORME | NUMBER | 1 | Yes | | |
| USUARIRESPONSABLE | ROWID | 10 | No | | |
| INTERVENCIOANTERIOR | NUMBER | 6 | Yes | | |
| NUMINTERVENCIONS | NUMBER | 0 | No | | |

**Taula Notes**

| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| IDNOTA | NUMBER | 0 | No | | |
| IDUSUARI | NUMBER | 0 | No | | |
| NOTA | VARCHAR2 | 1000 | Yes | | |
| NUMNOTA | NUMBER | 0 | No | | |

**Taula Responsables**

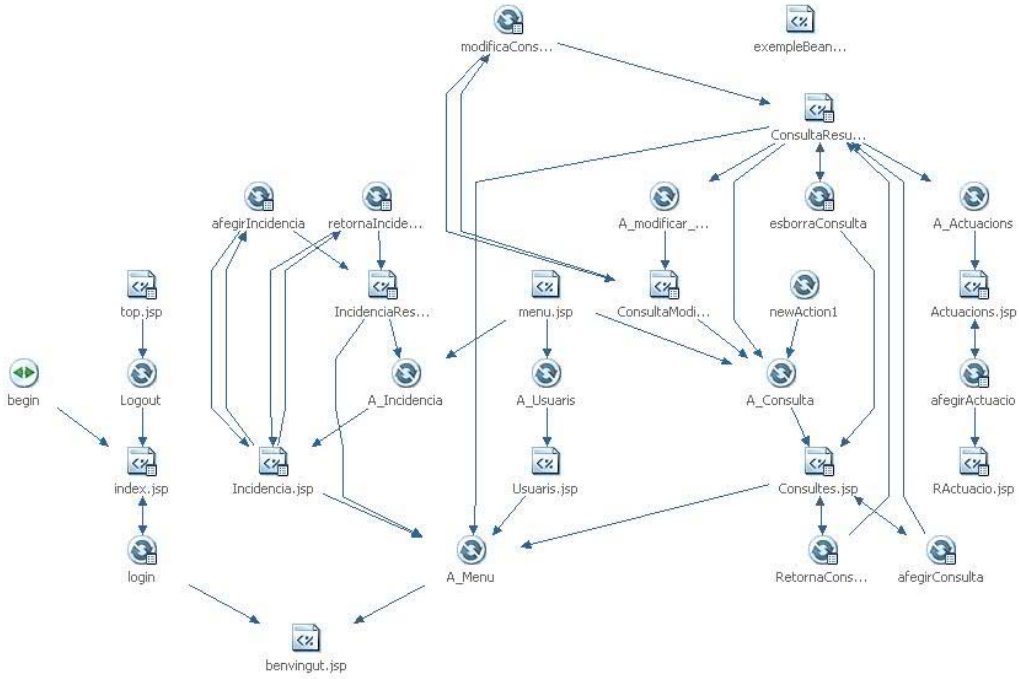
| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| IDRESPONSABLE | NUMBER | 0 | No | | |
| NOMRESPONSABLE | VARCHAR2 | 80 | No | | |
| CORREU | VARCHAR2 | 100 | No | | |
| AREA | VARCHAR2 | 30 | No | | |

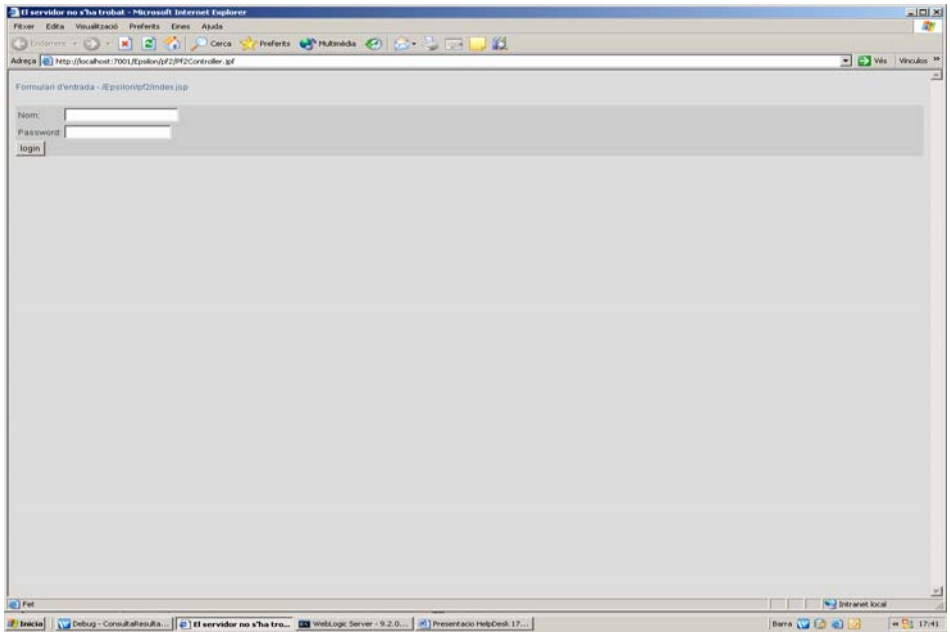
**Taula Usuaris**

| Nom del camp | Tipus de dades | Long. | Nuls? | Valor Inicial | Descripció |
|---|---|---|---|---|---|
| IDUSUARI | NUMBER | 10 | No | | |
| NOM | VARCHAR2 | 50 | No | | |
| CORREU | VARCHAR2 | 70 | No | | |
| PASSWORD | VARCHAR2 | 10 | No | | |
| TIPUSUSUARI | NUMBER | 1 | No | | |

## 3 CAPTURES DE PANTALLA

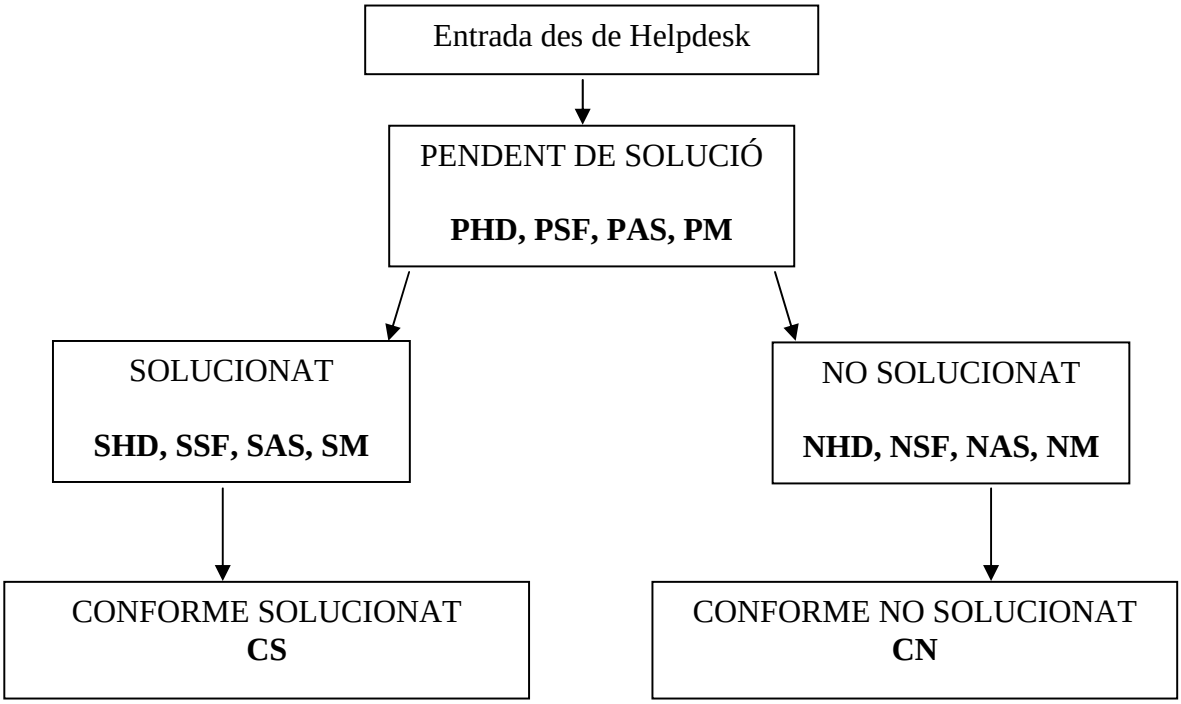
En fase de desenvolupament, mostrem les primeres imatges. No obstant, no són definitives, però permeten veure les funcionalitats i el mapa generat.



## 4 TAULA D'ESTATS I DESCRIPCIÓ

**CN:** Conforme No Solucionat
**CS:** Conforme Solucionat
**NAS:** No solucionat Area Sistemes
**NHD:** No solucionat HelpDesk
**NSF:** No solucionat Software
**NM:** No solucionat Manteniment
**PHD:** Pendent HelpDesk
**PSF:** Pendent Software
**PM:** Pendent Manteniment
**SAS:** Solucionat Area Sistemes
**SHD:** Solucionat HelpDesk
**SSF:** Solucionat Software
**SM:** Solucionat Manteniment



## 5 ASPECTES PER DEFINIR

Temes a concretar per part del dissenyador.

- Taules, camps i tipus de contingut dels camps.
- Hardware i software sobre el que correrà l’aplicació
- Numero de operaris
- Valoracions en aspectes com la accessibilitat i la facilitat d'ús.
- Necessitats de configuració i adaptabilitat

## 11.2 Segona visita.

**INDEX**

## INTRODUCCIÓ

La demostració que presentem és la versió de prova de l'aplicació Web Helpdesk. Aquesta versió és funcional en gran part. Les funcions que falten per completar són :

- Representació Gràfica dels Informes i la seva confecció.
- Servei de correu electrònic
- Entorn gràfic

L'aplicació s'ha instal·lat sobre un PC portàtil amb els servidors corresponents, de manera que aquesta té garantit l'escenari necessari per desenvolupar-se

**DESCRIPCIÓ**

L'aplicació Web té la següent estructura de funcionament : S'inicia entrant amb un navegador Web, en el nostre cas, Mozilla Firefox. La pàgina inicial correspon al formulari d'identificació. S'introdueix el nom i el password, i s'accedeix a la pàgina principal. Està formada per una zona superior, un menú i la pàgina de benvinguda, que ens comunica el nombre de correus nous rebuts, i l'estat del tancament automàtic de consultes.

La zona superior conté una imatge de fons i el títol. El títol conté un link al menú principal. A sota, trobem un vincle que ens tancarà la sessió i tornarem al formulari d'identificació. El menú ofereix els botons d'accés directe a les seccions de l'aplicació : Consultes, Incidències, Intervencions, Peticions, Informes, Correus i Opcions.

La pàgina Consultes ens ofereix diversos formularis, aquests són: Buscar per ID, buscar per usuari responsable, buscar per data, buscar per estat o bé afegir una nova consulta. El resultat d'una cerca ens dura a una pàgina de resultats. Des d'aquesta pàgina podrem modificar la consulta, esborrar-la o bé passar-la a Incidències.

Les Incidències són gestionades d'una manera similar. Tenim tants formularis com en Consultes, apart del formulari de nova incidència. Si busquem una incidència i anem a modificar-la, podrem veure que se'ns permet afegir una Intervenció o bé, veure les Intervencions associades a aquesta Incidència.

A la secció de Intervencions podem buscar Intervencions segons el Cap de Departament, la area, l'estat o bé la data d'intervenció. Des d'aquesta secció no podem afegir cap intervenció ja que aquestes estan vinculades a incidències o bé a peticions.

La secció d'informes possibilita veure el resum segons tres seleccions : Un informe anual complet, on es reflecteixen consultes, incidències i peticions durant l'any natural actual. El informe mensual ens mostra les incidències durant aquest mes natural, de la mateixa manera que el informe mensual complet, tot i que en aquest també s'inclouen les consultes i peticions del darrer mes.

Correu engloba un simple subsistema de correu, que ens permet enviar correus electrònics o bé rebre'ls.

Finalment, la secció Opcions ens permet configurar l'aplicació en els següents aspectes : Usuaris, Àrees, Estats, Notes i Responsables de Departament.

En aquesta versió no definitiva cal remarcar les mancances conegudes :

- Falta implementar un apartat que representi gràficament els informes generats.
- No incorpora control d'errors. Per exemple, si al camp Data introduïm una paraula, el programa retornarà un error general, sense especificar on es l'error.
- Dins la secció de correus, al veure els correus rebuts, l'aplicació es disposa a baixar tots els correus de la bústia, junt amb els seus adjunts, el que fa que l'aplicació es bloquegi durant un temps prolongat.
- L'informe mensual d'incidències no distingeix entre incidències de primer i segon nivell.

**TAULES DE DADES :**

| | Table | Tablespace | Rows |
|---|---|---|---|
| No | AREAS | DADESHELPDESK | 2 |
| No | CONFIG | DADESHELPDESK | |
| No | CONSULTA | DADESHELPDESK | 7 |
| No | ESTAT | DADESHELPDESK | 14 |
| No | INCIDENCIA | DADESHELPDESK | 4 |
| No | INFORMESAC | DADESHELPDESK | |
| No | INFORMESMC | DADESHELPDESK | |
| No | INFORMESMI | DADESHELPDESK | |
| No | INTERVENCIONSINCIDENCIA | DADESHELPDESK | 3 |
| No | INTERVENCIONSPETICIO | DADESHELPDESK | 0 |
| No | NOTES | DADESHELPDESK | 5 |
| No | PETICIONS | DADESHELPDESK | 1 |
| No | RESPONSABLESAREAS | DADESHELPDESK | 2 |
| No | USUARIS | DADESHELPDESK | 2 |

Taula AREAS

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| IDAREA | | NUMBER | 20 | 0 | No | No | | | |
| NOMAREA | | VARCHAR2 | 30 | 0 | No | No | | | |
| SITUACIO | | VARCHAR2 | 30 | 0 | No | No | | | |
| TELEFON | | VARCHAR2 | 30 | 0 | No | No | | | |
| CORREU | | VARCHAR2 | 30 | 0 | No | No | | | |
| DIRECTOR | | VARCHAR2 | 30 | 0 | No | No | | | |

Taula CONFIG

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| CAMP | | VARCHAR2 | 20 | 0 | No | No | | | |
| VALOR | | VARCHAR2 | 300 | 0 | No | No | | | |

Taula CONSULTA

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| NOM | | VARCHAR2 | 40 | 0 | No | No | | | |
| COGNOMS | | VARCHAR2 | 60 | 0 | No | No | | | |
| CORREU | | VARCHAR2 | 100 | 0 | No | Yes | | | |
| TELEFON | | VARCHAR2 | 15 | 0 | No | Yes | | | |
| DATACOMUNICACIO | | DATE | 7 | 0 | No | No | | | |
| TIPOLOGIACONSULTA | | NUMBER | 0 | 0 | No | No | | | |
| CONSULTA | | VARCHAR2 | 1000 | 0 | No | No | | | |
| SOLUCIOFINAL | | VARCHAR2 | 1000 | 0 | No | Yes | | | |
| ESTAT | | VARCHAR2 | 3 | 0 | No | No | | | |
| DATACONFORME | | DATE | 7 | 0 | No | Yes | | | |
| USUARIRESPONSABLE | | VARCHAR2 | 10 | 0 | No | Yes | | | |
| IDCONSULTA | | NUMBER | 0 | 0 | No | No | | | |

Taula ESTAT

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| IDESTAT | | VARCHAR2 | 3 | 0 | No | No | | | |
| DESCRIPCIO | | VARCHAR2 | 30 | 0 | No | No | | | |

Taula INCIDENCIA

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| NOM | | VARCHAR2 | 40 | 0 | No | No | | | |
| COGNOMS | | VARCHAR2 | 60 | 0 | No | No | | | |
| CORREU | | VARCHAR2 | 100 | 0 | No | Yes | | | |
| TELEFON | | VARCHAR2 | 15 | 0 | No | Yes | | | |
| DATACOMUNICACIO | | DATE | 7 | 0 | No | No | | | |
| INCIDENCIA | | VARCHAR2 | 1000 | 0 | No | No | | | |
| SOLUCIOFINAL | | VARCHAR2 | 1000 | 0 | No | Yes | | | |
| ESTAT | | VARCHAR2 | 3 | 0 | No | No | | | |
| DATACONFORME | | DATE | 7 | 0 | No | Yes | | | |
| USUARIRESPONSABLE | | VARCHAR2 | 10 | 0 | No | No | | | |
| AREA | | VARCHAR2 | 20 | 0 | No | No | | | |
| IDINCIDENCIA | | NUMBER | 10 | 0 | No | No | | | |
| TIPOLOGIAINCIDENCIA | | NUMBER | 10 | 0 | No | No | | | |
| NUMINTERVENCIO | | NUMBER | 0 | 0 | No | No | | | |

Taula INFORME A C

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| NUMINCIDENCIES | | NUMBER | 0 | 0 | No | No | | | |
| NUMPETICIONS | | NUMBER | 0 | 0 | No | No | | | |
| NUMCONSULTES | | NUMBER | 0 | 0 | No | No | | | |
| IDINFORME | | NUMBER | 0 | 0 | No | No | | | |
| DATA | | DATE | 7 | 0 | No | No | | | |

Taula INFORME M C

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| NUMINCIDENCIES | | NUMBER | 0 | 0 | No | No | | | |
| NUMPETICIONS | | NUMBER | 0 | 0 | No | No | | | |
| NUMCONSULTES | | NUMBER | 0 | 0 | No | No | | | |
| IDINFORME | | NUMBER | 0 | 0 | No | No | | | |
| DATA | | DATE | 7 | 0 | No | No | | | |

Taula INFORME M I

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| NUMINCIDENCIES1 | | NUMBER | 0 | 0 | No | No | | | |
| NUMINCIDENCIES2 | | NUMBER | 0 | 0 | No | No | | | |
| IDINFORME | | NUMBER | 0 | 0 | No | No | | | |
| DATA | | DATE | 7 | 0 | No | No | | | |

Taula INTERVENCIO - INCIDENCIA

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| IDINTERVENCIO | | NUMBER | 10 | 0 | No | No | | | |
| DATAINICI | | DATE | 7 | 0 | No | No | | | |
| TIPUSINTERVENCIO | | NUMBER | 1 | 0 | No | No | | | |
| ESTAT | | VARCHAR2 | 3 | 0 | No | No | | | |
| AREA | | VARCHAR2 | 20 | 0 | No | Yes | | | |
| PERSONAAREA | | VARCHAR2 | 20 | 0 | No | No | | | |
| DESCRIPCIO | | VARCHAR2 | 1000 | 0 | No | No | | | |
| SOLUCIO | | VARCHAR2 | 1000 | 0 | No | No | | | |
| DATASOLUCIO | | DATE | 7 | 0 | No | Yes | | | |
| INTERVENCIOANTERIOR | | NUMBER | 10 | 0 | No | Yes | | | |
| NUMINTERVENCIONS | | NUMBER | 10 | 0 | No | Yes | | | |
| IDINCIDENCIA | | NUMBER | 0 | 0 | No | No | | | |

Taula INTERVENCIO - PETICIO

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| IDINTERVENCIO | | NUMBER | 10 | 0 | No | No | | | |
| DATAINICI | | DATE | 7 | 0 | No | No | | | |
| TIPUSINTERVENCIO | | NUMBER | 1 | 0 | No | No | | | |
| ESTAT | | VARCHAR2 | 3 | 0 | No | No | | | |
| AREA | | VARCHAR2 | 20 | 0 | No | Yes | | | |
| PERSONAAREA | | VARCHAR2 | 20 | 0 | No | No | | | |
| DESCRIPCIO | | VARCHAR2 | 1000 | 0 | No | No | | | |
| SOLUCIO | | VARCHAR2 | 1000 | 0 | No | No | | | |
| DATASOLUCIO | | DATE | 7 | 0 | No | Yes | | | |
| INTERVENCIOANTERIOR | | NUMBER | 10 | 0 | No | Yes | | | |
| NUMINTERVENCIONS | | NUMBER | 10 | 0 | No | No | | | |
| IDPETICIO | | NUMBER | 10 | 0 | No | No | | | |

Taula NOTES

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| IDNOTA | | NUMBER | 10 | 0 | No | No | | | |
| IDUSUARI | | NUMBER | 10 | 0 | No | No | | | |
| NOTA | | VARCHAR2 | 1000 | 0 | No | Yes | | | |
| NUMNOTA | | NUMBER | 10 | 0 | No | No | | | |
| TITOL | | VARCHAR2 | 25 | 0 | No | No | | | |

Taula PETICIO

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| IDPETICIO | | NUMBER | 10 | 0 | No | No | | | |
| DATACOMUNICACIO | | DATE | 7 | 0 | No | No | | | |
| TIPOLOGIAPETICIO | | NUMBER | 10 | 0 | No | No | | | |
| PETICIO | | VARCHAR2 | 1000 | 0 | No | No | | | |
| ESTAT | | VARCHAR2 | 3 | 0 | No | No | | | |
| USUARIRESPONSABLE | | VARCHAR2 | 10 | 0 | No | No | | | |
| AREA | | VARCHAR2 | 20 | 0 | No | No | | | |
| DATACONFORME | | DATE | 7 | 0 | No | Yes | | | |
| SOLUCIOFINAL | | VARCHAR2 | 1000 | 0 | No | Yes | | | |
| RESPONSABLEAREA | | VARCHAR2 | 20 | 0 | No | No | | | |
| NUMINTERVENCIO | | NUMBER | 0 | 0 | No | No | | | |

Taula RESPONSABLES

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| IDRESPONSABLE | | NUMBER | 10 | 0 | No | No | | | |
| NOMRESPONSABLE | | VARCHAR2 | 80 | 0 | No | No | | | |
| CORREU | | VARCHAR2 | 100 | 0 | No | No | | | |
| AREA | | VARCHAR2 | 30 | 0 | No | No | | | |

Taula USUARIS

| Name | Schema | Datatype | Size | Scale | Ref | Nulls? | Default Value | Scope Schema | Scope Table |
|---|---|---|---|---|---|---|---|---|---|
| IDUSUARI | | NUMBER | 0 | 0 | No | No | | | |
| NOM | | VARCHAR2 | 50 | 0 | No | No | | | |
| CORREU | | VARCHAR2 | 70 | 0 | No | No | | | |
| PASSWORD | | VARCHAR2 | 10 | 0 | No | No | | | |
| TIPUSUSUARI | | NUMBER | 1 | 0 | No | No | | | |

# 11.3 Documentació en l’entrega de l’aplicatiu. 18 de Juliol de 2007

**ÍNDEX**

## *1. ANÀLISI FUNCIONAL DE L'APLICATIU.*

•**Enregistrament, Cerca, Modificació i Eliminació de incidències i intervencions.** Mitjançant una sèrie de senzills formularis podem accedir als formularis que ens permeten introduir les incidències a la base de dades. De la mateixa manera, i a la mateixa pàgina, disposem d'un formulari per cercar les incidències en funció de diversos criteris. Un cop seleccionada la incidència entre els resultats podrem modificar les dades, o bé esborrar-la.

•**Compartició informació a través del bloc de notes.** Per facilitar les tasques de resolució d'incidències, s'ha previst un senzill sistema de notes, de manera que els diferents usuaris de l'aplicatiu poden compartir informació, a través de notes, identificades amb un nombre i un títol, per facilitar la seva localització.

•**Llistats i representacions gràficament les incidències per períodes.** L'aplicatiu permet realitzar llistats de les activitats enregistrades en funció del període de temps : anual o bé mensual, i de les característiques del informe, segons sigui complet, o bé específic en funció de les intervencions. També es permeten obtenir gràfiques que representen les dades dels llistats anteriors. Aquestes gràfiques permeten una visualització ràpida i senzilla de l'activitat del servei.

•**Comunicació de les incidències entre àrees.** Per facilitar la comunicació de les intervencions a les àrees responsables d'actuar en cada cas, s'implementa l'enviament automàtic de correu electrònic a les bústies de les àrees. D'aquesta manera, s'informa immediatament de les activitats en el sentit del servei de Helpdesk cap a les àrees d'intervenció.

•**Generació d'informes automàtics en format RTF.** Per altra banda, quan s'informa d'una incidència, es genera un informe destinat al usuari, per poder registrar l'atenció per part del servei de Helpdesk. Aquest informe s'envia automàticament via correu electrònic a l'adreça de correu indicada en el moment del registre de la incidència.

## 2. DESCRIPCIÓ DELS APARTATS DE L'APLICATIU

En aquest apartat, donarem una petita volta per l'aplicatiu per explicar breument les funcions que ens brinda l'aplicatiu.

Quan accedim a l'aplicatiu a través del link que disposem a l'escriptori, el primer que ens trobem és el formulari d'autenticació. Per defecte, el programa s'entrega amb dos usuaris introduïts per defecte, un és l'administrador, en *Bruce*, i l'altre és un operador, en *Jackie*. En *Bruce* té accés a modificar les opcions de l'aplicació, al contrari que en *Jackie*. Per accedir com a *Bruce*, introduirem a nom el text : "*Bruce*" i com a contrasenya "*Lee*". Per accedir com a *Jackie*, introduirem "*Jackie*" i "*Chan*".

Amb qualsevol dels dos accedirem al menú principal, des del qual podrem accedir a les seccions en les que es divideix l'aplicatiu. En la pàgina del menú principal podem veure també el nombre de correus rebuts i no llegits, d'acord amb la bústia de correu per defecte. En aquest cas, s'ha introduït la bústia "*tfunez*", amb la configuració del servidor de correu POP3 *pop3.gencat.net*.

El primer lloc on anirem serà la secció Consultes. Si fem clic al botó on fica "Consultes" de la part esquerra de la barra principal, accedirem a la secció desitjada.

Podem veure que en un primer terme ens apareix un formulari que consta de diferents camps. Aquest camp serveix per trobar consultes dintre la base de dades. Per exemple, podem introduir el nom de la persona afectada, o bé, la data de comunicació, l'usuari responsable de la consulta o bé l'estat.

Més avall podrem veure el formulari d'introducció de consultes. Serveis per arxivar les consultes, un cop haguem introduït les dades correctament. Si no les emplenem bé, el software ens indicarà a través d'un missatge d'error el problema.

Tant si busquem com si afegim una consulta, si tenim èxit ens durà a una pàgina on es llista la/es consulta/es. Es mostren les dades més rellevants de la consulta. Si ens hi fixem, a la part dreta veurem una sèrie d'enllaços. Per aquest cas de consultes, són Passar a Incidència, Passar a Petició, Modificar i Esborrar.

Les consultes, en funció de la seva dificultat, les podem resoldre telefònicament, o bé elevar-les al nivell de Incidències o Peticions. D'aquesta manera, se'ns durà al formulari de nova incidència o petició, amb les dades corresponents a la consulta. D'aquesta manera ens serà més senzill completar la introducció de la nova incidència o petició.

Quan donem d'alta una incidència o petició i accedim a ella a través del sistema de cerca, podem accedir a modificar-la. A diferència de les consultes, en les incidències i peticions, es gestionen també les intervencions fetes per trobar la solució. Per a tal, sota el formulari de modificació, al que podem accedir a través de l'enllaç previst al llistat, podrem obrir el formulari de nova intervenció, o bé obtenir un llistat amb les intervencions ja introduïdes anteriorment.

Per últim, Consultes, Incidències i Peticions comparteixen el fet de que quan fem clic a Esborrar, quan són al llistat, l'aplicatiu ens demana la confirmació abans de procedir-hi.

L'altra secció disponible és la de Informes. Aquesta ens permet seleccionar el període i el tipus d'informe amb el que podrem extreure llistats de l'activitat recollida durant el període seleccionats. Des de l'obtenció del llistats, disposarem d'un botó que ens permetrà accedir a la representació gràfica del llistat en qüestió. El gràfic està basat en un diagrama de línies, que es dinàmic en funció dels resultats. Els conceptes en diferencien pel color i la forma dels punts.

Després podem accedir a la secció de Notes, en la qual podem anotar informació en un quadre de text senzill. Un cop guardat, aquesta informació serà visible per a tots els usuaris. D'aquesta manera, es facilita la comunicació entre persones del mateix servei. S'utilitza per compartir metodologia, respostes freqüents, etc.

I finalment podem accedir a les Opcions, sols si tenim drets d'administrador, per tant, si som operadors, no podrem accedir-hi. Dintre opcions, es troben els següents subseccions :

- **Usuaris** : Ens permet gestionar el nombre i el tipus d'usuaris que s'emmagatzemen en la base de dades.

- **Estats** : Es poden definir els estats en que es pot trobar una incidència, com per exemple, Solucionat, No Solucionat, Pendent de Solució, etc.

- **Correu** : Dintre aquesta subsecció es troben les dades de configuració pròpies del sistema de correu. Quan s'hi accedeix les dades no es mostren en pantalla per seguretat. En el millor dels casos, aquesta informació s'introdueix en instal·lar l'aplicatiu i no es torna a variar.

- **Tancar Any** : S'utilitza per gestionar les incidències que pertanyen a un exercici anterior. En el moment que s'actualitza aquesta informació, les incidències anterior a aquest any, no apareixeran en la cerca comú d'incidències, si en canvi, en el cas dels informes.

- **Unitats Administratives** : Aquí es defineix l'estructura de la entitat a la qual pertany la xarxa informàtica a la qual es dona servei a través de Helpdesk. Aquesta pàgina permet afegir o esborrar unitats a l'entitat.

- **Responsables de les Unitats Administratives** : D'acord amb el funcionament de les incidències comunicades pels caps de les Unitats Administratives, es guarda un registre de cadascun de les persones responsables d'emetre peticions.

- **Tipologies** : Conjunt de conceptes que defineix la temàtica de les incidències que s'enregistren. Per exemple, podrien ser "Fallada elèctrica", o "Atac de virus".

- **Àrees** : En aquest apartat podrem donar d'alta o esborrar les àrees d'actuació per part del Helpdesk, com poden ser Maquinari, Programari,etc .

- **Responsables de les Àrees** : Per a l'actuació de les persones encarregades dintre de les àrees de realitzar les intervencions, es generen una sèrie de comunicacions que s'automatitzen a través de la introducció de les dades a la base de dades.

## 3. REPRESENTACIÓ DE LES TAULES DE DADES

### *3.1. Taula AREES*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| IDAREA | NÚMERIC | 0 | No |
| NOMAREA | ALFANUMÉRIC | 30 | No |
| SITUACIO | ALFANUMÉRIC | 30 | No |
| TELEFON | ALFANUMÉRIC | 30 | No |
| CORREU | ALFANUMÉRIC | 30 | No |
| DIRECTOR | ALFANUMÉRIC | 30 | No |

### *3.2. Taula CONFIG*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| CAMP | ALFANUMÉRIC | 20 | No |
| VALOR | ALFANUMÉRIC | 500 | No |

### *3.3. Taula CONSULTES*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| NOM | ALFANUMÉRIC | 40 | No |
| COGNOMS | ALFANUMÉRIC | 60 | No |
| CORREU | ALFANUMÉRIC | 100 | Sí |
| TELEFON | ALFANUMÉRIC | 15 | Sí |
| DATACOMUNICACIO | DATA | 7 | No |
| CONSULTA | ALFANUMÉRIC | 1000 | No |
| SOLUCIOFINAL | ALFANUMÉRIC | 1000 | Sí |
| ESTAT | ALFANUMÉRIC | 30 | No |
| DATACONFORME | DATA | 7 | Sí |
| USUARIRESPONSABLE | ALFANUMÉRIC | 10 | Sí |
| IDCONSULTA | NÚMERIC | 0 | No |
| TIPOLOGIA | ALFANUMÉRIC | 20 | Sí |
| UNITATADMINISTRATIVA | ALFANUMÉRIC | 30 | Sí |
| CODI | ALFANUMÉRIC | 15 | Sí |

### *3.4. Taula ESTATS*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| IDESTAT | ALFANUMÉRIC | 3 | No |
| DESCRIPCIO | ALFANUMÉRIC | 30 | No |
| TANCAT | ALFANUMÉRIC | 10 | No |

### *3.5. Taula INCIDÈNCIES*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| NOM | ALFANUMÉRIC | 40 | No |
| COGNOMS | ALFANUMÉRIC | 60 | No |
| CORREU | ALFANUMÉRIC | 100 | Sí |
| TELEFON | ALFANUMÉRIC | 15 | Sí |
| DATACOMUNICACIO | DATA | 7 | No |
| INCIDENCIA | ALFANUMÉRIC | 1000 | No |
| SOLUCIOFINAL | ALFANUMÉRIC | 1000 | Sí |
| ESTAT | ALFANUMÉRIC | 30 | No |
| DATACONFORME | DATA | 7 | Sí |
| USUARIRESPONSABLE | ALFANUMÉRIC | 10 | No |
| IDINCIDENCIA | NÚMERIC | 10 | No |
| NUMINTERVENCIO | NÚMERIC | 0 | No |
| ADJUNT | ALFANUMÉRIC | 50 | Sí |
| TIPOLOGIA | ALFANUMÉRIC | 20 | Sí |
| UNITAT ADMINISTRATIVA | ALFANUMÉRIC | 30 | Sí |
| CODI | ALFANUMÉRIC | 15 | Sí |

### *3.6. Taula INTERVENCIONS*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| IDINTERVENCIO | NÚMERIC | 10 | No |
| DATAINICI | DATA | 7 | No |
| TIPUSINTERVENCIO | ALFANUMÉRIC | 10 | No |
| ESTAT | ALFANUMÉRIC | 30 | No |
| AREA | ALFANUMÉRIC | 20 | Sí |
| PERSONAAREA | ALFANUMÉRIC | 20 | No |
| DESCRIPCIO | ALFANUMÉRIC | 1000 | No |
| INTERVENCIOANTERIOR | NÚMERIC | 10 | Sí |
| ADJUNT | ALFANUMÉRIC | 60 | Sí |
| ORIGEN | ALFANUMÉRIC | 1 | Sí |
| ID | NÚMERIC | 0 | No |

### *3.7. Taula NOTES*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| IDNOTA | NÚMERIC | 10 | No |
| TITOL | ALFANUMÉRIC | 50 | No |
| NOTA | ALFANUMÉRIC | 1000 | Sí |

### *3.8. Taula PETICIONS*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| IDPETICIO | NÚMERIC | 10 | No |
| DATACOMUNICACIO | DATA | 7 | No |
| PETICIO | ALFANUMÉRIC | 1000 | No |
| ESTAT | ALFANUMÉRIC | 30 | No |
| USUARIRESPONSABLE | ALFANUMÉRIC | 10 | No |
| DATACONFORME | DATA | 7 | Sí |
| SOLUCIOFINAL | ALFANUMÉRIC | 1000 | Sí |
| NUMINTERVENCIO | NÚMERIC | 0 | No |
| ADJUNT | ALFANUMÉRIC | 50 | Sí |
| UNITATADMINISTRATIVA | ALFANUMÉRIC | 20 | Sí |
| TIPOLOGIA | ALFANUMÉRIC | 30 | Sí |
| RESPONSABLEUA | ALFANUMÉRIC | 10 | No |
| PERSONACONTACTE | ALFANUMÉRIC | 10 | Sí |
| CODI | ALFANUMÉRIC | 15 | Sí |

### *3.9. Taula RESPONSABLES AREES*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| IDRESPONSABLE | NÚMERIC | 10 | No |
| NOMRESPONSABLE | ALFANUMÉRIC | 80 | No |
| CORREU | ALFANUMÉRIC | 100 | No |
| AREA | ALFANUMÉRIC | 30 | No |

### *3.10. Taula RESPONSABLES UNITAT ADMINISTRATIVA*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| IDRESPONSABLE | NÚMERIC | 10 | No |
| NOMRESPONSABLE | ALFANUMÉRIC | 80 | No |
| CORREU | ALFANUMÉRIC | 100 | No |
| UA | ALFANUMÉRIC | 30 | No |

### *3.11. Taula TIPOLOGIES*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| NOM | ALFANUMÉRIC | 15 | No |
| DESCRIPCIO | ALFANUMÉRIC | 500 | No |

### *3.12. Taula UNITATS ADMINISTRATIVES*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| NOM | ALFANUMÉRIC | 30 | No |
| DIRECCIO | ALFANUMÉRIC | 50 | No |
| TELEFON | ALFANUMÉRIC | 12 | No |
| CORREU | ALFANUMÉRIC | 30 | No |
| RESPONSABLE | ALFANUMÉRIC | 30 | No |
| IDUA | NÚMERIC | 0 | Sí |

### *3.13. Taula USUARIS*

| Nom | Tipus de dades | Mida | Valors Buits? |
|---|---|---|---|
| IDUSUARI | NÚMERIC | 0 | No |
| NOM | ALFANUMÉRIC | 50 | No |
| CORREU | ALFANUMÉRIC | 70 | No |
| PASSWORD | ALFANUMÉRIC | 10 | No |
| TIPUSUSUARI | NÚMERIC | 38 | No |

# 11.4 Manual d’usuari

**ÍNDEX**

## I. INTRODUCCIÓ

L'Aplicatiu HELPDESK té com a finalitat cobrir les necessitats del Departament de Informàtica en gestió de incidències d'una xarxa informàtica.

L'estructura del funcionament del servei de Helpdesk és la següent :

Les incidències ocorregudes en l'ús dels serveis de la xarxa són comunicades al servei Helpdesk via telèfon o correu electrònic. Un cop rebudes les incidències, aquestes es classifiquen segons la seva naturalesa:

**CONSULTES** : Són incidències de importància lleu, la resposta o solució es pot tramitar de manera immediata, i s'espera una resolució també immediata. Per tant, no requereixen de la intervenció del Departament d'Informàtica.
Les dades recollides són les següents :

Codi
Nom
Cognoms
Correu
Telèfon
Data de Comunicació
Consulta
Solució Final
Estat
Data de Conformitat
Usuari Responsable
Tipologia
Unitat Administrativa

Un cop resolta la consulta, s'arxiva i queda enregistrada al sistema.

**INCIDÈNCIES** : Són incidències d'una importància lleu o greu, en que la solució necessita de la intervenció del Departament d'Informàtica. Per a enregistrar aquestes, es deriven en el que s'anomenen intervencions. Les dades requerides per caracteritzar una incidència són les següents :

Codi
Nom
Cognoms
Correu
Telèfon,
Data de Comunicació
Incidència
Solució Final
Estat
Data de Conformitat
Usuari Responsable
Número d'Intervencions
Adjunt
Tipologia
Unitat Administrativa

**PETICIONS** : Les peticions són incidències de importància variable, que tenen la particularitat de ser tramitades pels caps de Unitat, de manera que aquests disposen de un canal preferent per tramitar

les incidències. Aquestes peticions contemplen la necessitat de realitzar intervencions també. Per a enregistrar les peticions, són necessàries les següents dades:

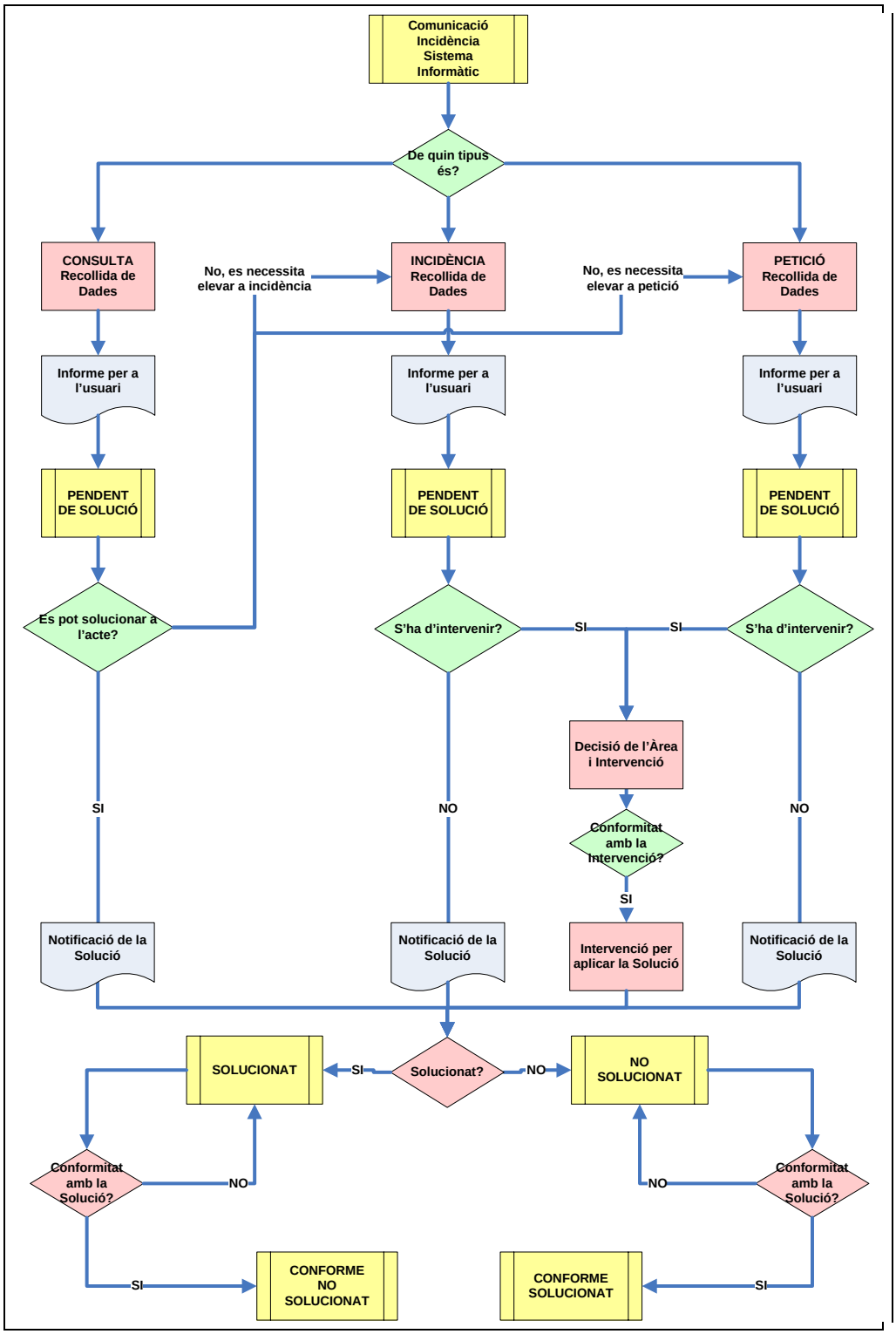
Codi
Persona de Contacte
Responsable
Data de Comunicació
Petició
Estat
Usuari Responsable
Data de Conformitat
Solució Final
Número d'Intervencions
Adjunt
Unitat Administrativa
Tipologia

**INTERVENCIONS** : Les intervencions són les dades que enregistren les operacions realitzades pel personal del Departament d'Informàtica sobre la incidència o la petició. Aquestes venen caracteritzades pels camps següents :

Data d'Inici
Tipus d'Intervenció
Estat
Area
Persona d'àrea
Descripció
Solució
Data de Solució
Intervenció Anterior
Adjunt
Codi Incidència
Num. d'Intervencions

El procés que segueixen les incidències a la xarxa informàtica, segueixen el següent diagrama de flux, i d'aquesta manera s'ha implementat a l'aplicatiu. Les incidències arriben i s'avaluen en funció de la seva solució. A partir d'aquí es proposa una solució i s'aplica. S'avalua si la incidència ha estat resolta, i si l'usuari està conforme amb el resultat.

Aquest diagrama de flux de les incidències dóna lloc als següents estats en els que es pot trobar la incidència en el procés de resolució. Aquests estats són els que es mostraran a continuació.


Comunicació Incidència Sistema Informàtic
De quin tipus és?
CONSULTA Recollida de Dades
No, es necessita elevar a incidència
INCIDÈNCIA Recollida de Dades
No, es necessita elevar a petició
PETICIÓ Recollida de Dades
Informe per a l'usuari
Informe per a l'usuari
Informe per a l'usuari
PENDENT DE SOLUCIÓ
PENDENT DE SOLUCIÓ
PENDENT DE SOLUCIÓ
Es pot solucionar a l'acte?
S'ha d'intervenir?
SI
SI
S'ha d'intervenir?
Decisió de l'Àrea i Intervenció
SI
NO
Conformitat amb la Intervenció?
NO
SI
Notificació de la Solució
Notificació de la Solució
Intervenció per aplicar la Solució
Notificació de la Solució
SOLUCIONAT
SI
Solucionat?
NO
NO SOLUCIONAT
Conformitat amb la Solució?
NO
NO
Conformitat amb la Solució?
SI
CONFORME NO SOLUCIONAT
CONFORME SOLUCIONAT
SI

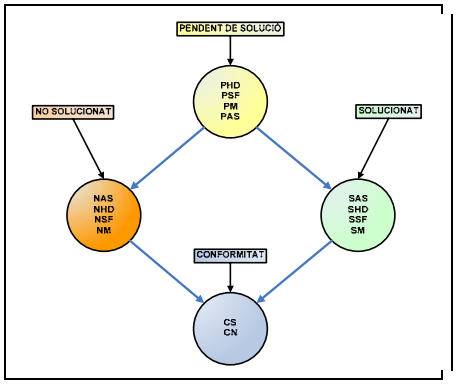
**ESTATS DE LES INCIDENCIES**

**PHD** : Pendent de Solució Helpdesk
**PAS** : Pendent de Solució Àrea Sistemes
**PSF** : Pendent de Solució Software
**PM** : Pendent de Solució Manteniment

**NHD** : No Solucionat Helpdesk
**NAS** : No Solucionat Àrea Sistemes
**NSF** : No Solucionat Software
**NM** : No Solucionat Manteniment

**SHD** : Solucionat Helpdesk
**SAS** : Solucionat Àrea Sistemes
**SSF** : Solucionat Software
**SM** : Solucionat Manteniment

**CN** : Conforme No Solucionat
**CS** : Conforme Solucionat




Aquest aplicatiu, per a coordinar les accions entre les diferents parts implicades, utilitza un sistema de correus electrònics que permet saber al afectat l'estat de la seva incidència. A més, també s'utilitza per comunicar a les àrees del Departament d'Informàtica de les accions necessàries i les incidències ocorregudes.

Per a l'anàlisi de les incidències de la xarxa informàtica l'aplicatiu permet obtenir informes amb el nombre d'incidències enregistrades durant un mes natural o bé l'any natural. Aquestes dades també poden ser visualitzades en forma de gràfics.

L'aplicatiu a més, disposa d'una gestió d'usuaris que permet establir perfils de control per oferir permisos en la gestió de l'aplicatiu. Aquests permisos permeten configurar el sistema de correu, les dades disponibles i el comportament del propi aplicatiu. Mitjançant nom d'usuari i password, els usuaris s'autentifiquen i accedeixen al sistema, amb el perfil d'Administrador o Operatiu. L'Administrador té la capacitat de configurar l'aplicatiu, mentre que l'Operatiu sols pot gestionar les incidències i obtenir informes.

Donada la problemàtica de que algunes incidències romanen sense solució o bé sense conformitat, pel motiu que sigui, s'ha implementat un sistema de auto-tancament, en funció d'un nombre de dies predeterminat.

## II.- CONSULTES.

A les consultes s'accedeix mitjançant el botó "Consultes" del menú principal. La pàgina a la que ens condueix està dividida en dues parts, cerca de consultes i formulari de introducció de consultes.

### II.I.- Cerca de consultes.

La cerca de consultes es realitza mitjançant un senzill formulari, que es basa en els següents camps : Codi de la consulta, Usuari Responsable de la Consulta, Cognoms de l'usuari afectat, Data de comunicació i Unitat Administrativa. El formulari de cerca realitza la cerca en funció de un dels paràmetres. El resultat de la cerca es retorna en una pàgina on es llisten els resultats coincidents. En aquesta pàgina s'enumeren les següents dades : Codi, Estat, Data de Comunicació, Nom, Cognoms, Telèfon, Correu electrònic, Unitat Administrativa, Usuari Responsable, Tipologia, Text de la consulta, Data de Conformitat i Solució Final. Per a cadascuna de les consultes retornades, existeixen quatre enllaços per a diferents accions: Modificar, Esborrar, Passar a Incidència i Passar a Petició.

L'acció de Modificar ens porta directament a una pàgina que sen permet modificar totes les dades de la consulta excepte el codi. A més, podem afegir la Solució Final i la Data de Conformitat. Quan una consulta s'estableix com a tancada, ja no es podrà modificar.

### II.II.- Introducció de Consultes.

Les Consultes són introduïdes a través del formulari present en la pàgina. En aquest se'ns demana la següent informació :

Data de Comunicació (s'ha d'introduir en el següent format: DD/MM/AAAA)

**III.- INCIDÈNCIES.**

A la pàgina d'Incidències s'accedeix a través del botó del mateix nom, situat al menú principal. La pàgina d'Incidències es divideix en cerca i introducció.

La cerca es pot realitzar a través dels següents criteris :

Un cop seleccionada la incidència, podem afegir o veure les intervencions derivades a través de l'opció Modificar.

## IV.- PETICIONS.

Les peticions utilitzen el mateix sistema que les incidències, per tant, comparteixen metodologia i patrons.

La pàgina de peticions consta de formulari de cerca i formulari d'introducció de dades. El resultat de la cerca, ens permet modificar, esborrar, afegir i veure les intervencions.

## V.- OPCIONS

Les opcions estan només disponibles per al perfil d'administrador.

*Usuaris* : Permet crear, eliminar i modificar usuaris i els permisos que tenen. S'estableixen en dos perfils, Operatiu i Administrador. Com ja s'ha dit, l'administrador. Els administrador s'identifiquen per ser del tipus 500, de manera que qualsevol usuari amb un valor Tipus inferior, no tindrà accés a les opcions de configuració. Els usuaris venen caracteritzats per la ID, el Nom, el Correu, el Password i el Tipus d'Usuari.

*Àrees* : En aquest apartat es pot configurar les dades de les diferents àrees del Departament d'Informàtica en funció de la seva estructura operativa. Les accions possibles són les d'afegir, modificar i eliminar les àrees del Departament. Les àrees venen determinades per dades com el Nom, la Situació, el Telèfon, el Correu i el Director d' Àrea.

*Responsable d' Àrea* : El responsables d'Àrea representen les persones encarregades de rebre les notificacions sobre les intervencions a realitzar amb motiu d'una incidència. Per identificar-los s'utilitzen el Nom, la Àrea a la qual pertany i el Correu electrònic per contactar-hi.

*Estats* : De la mateixa manera, es poden modificar els estats en que pot estar una incidència. A més, els estats determinen si la incidència implica el tancament de la mateixa, el seu estat previ (susceptible a ésser tancat passat un espai de temps).

*Tancament de l'any vigent* : En aquest apartat, es permet tancar l'any vigent, de manera que les incidències registrades amb anterioritat ja no apareixeran.

*Unitat Administrativa*: Les unitats administratives configuren l'estructura de l'organització al servei de la qual està la secció de Helpdesk. Els camps que l'identifiquen són : Nom, Direcció, Telèfon, Correu, Responsable de la U.A. i ID

*Responsable Unitat Administrativa*: Formen part de les Unitats Administratives, i esdevenen subjectes de les Peticions. Per tant, les Peticions estan orientades cap aquestes persones. Són necessàries les següents dades:
Nom del Responsable, Correu i Unitat Administrativa.

*Tipologies* : Les tipologies són les etiquetes que permeten definir el tipus d'incidència d'una manera senzilla. Estan definides pel Nom i la Descripció de les mateixes.

*Configuració del correu*: Per configurar el sistema de correu electrònic s'han d'obtenir les següents dades :

- SMTP Host : Nom o adreça IP del servidor de correu sortint (SMTP).
- SMTP Port : Port TCP del servidor de correu sortint.
- POP Host : Nom o adreça IP del servidor de correu entrant (POP)
- POP Port : Port TCP del servidor de correu entrant.
- Nom d'usuari : Per a accedir al servei de correu electrònic, es necessita un nom d'usuari, generalment, l'adreça de correu que s'utilitza per al servei de e-mail.
- Password: El password s'utilitza per accedir al servei del correu electrònic juntament amb el nom d'usuari.
- Cos Petició: Text predeterminat que s'enviarà amb el correu electrònic autogenerat quan s'agregui una petició.
- Cos Incidència: Text predeterminat que s'enviarà amb el correu electrònic autogenerat quan s'agregui una Incidència.

- Ruta RTF : Camí fins la plantilla de l'informe RTF que s'adjunta automàticament quan s'agrega un incidència o bé una petició.
- Ruta Guardar: Camí fins a la carpeta on s'emmagatzemaran les dades provinents dels adjunts dels correus electrònics entrants.

## VI.- EXTRES

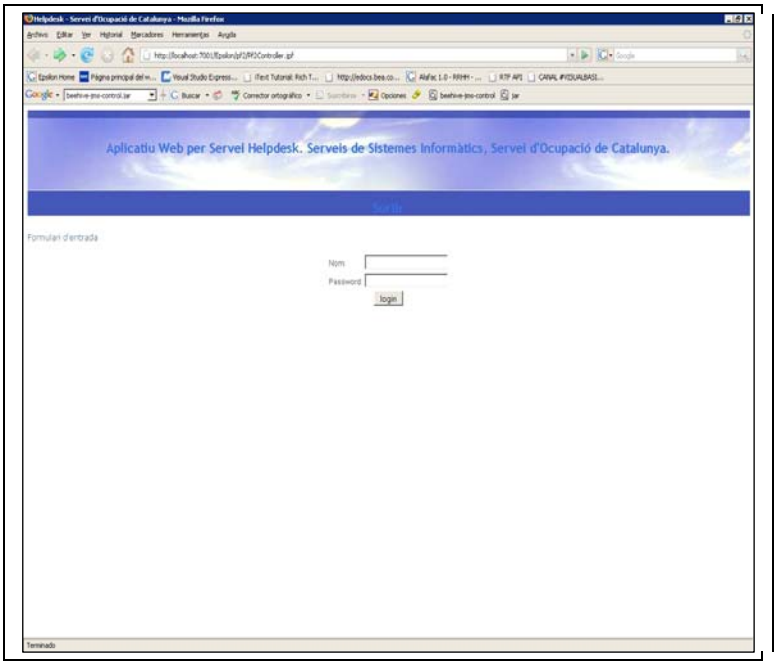
Gràfics : El sistema de generació de gràfics està integrat en la secció d'informes

**Notes**: El sistema de notes és un senzill sistema de compartició de dades de text entre els usuaris a través de l'aplicatiu. És accessible a través del botó homònim del menú principal. A continuació apareix un llistats de notes, ordenades per data d'introducció, amb el seu corresponent títol. Si fem clic en qualsevol d'elles, es recupera en un formulari que permet la seva modificació.

**Sistema de correu**: El sistema de correu, tot i ser utilitzat per l'aplicatiu per comunicar-se amb els diferents departaments de l'organització, permet rebre i enviar correus electrònics. Si fem clic, en la pantalla de benvinguda sobre el número de correus no llegits, passarem a un nou lloc, on podrem veure els correus rebuts, i seleccionar-los per poder veure'ls completament. L'altre botó "Enviar" ens porta a un formulari de redacció d'un correu electrònic amb les opcions de "Per a", "De", "Assumpte," i "Adjunt". El sistema de correu electrònic només permet enviar un fitxer, de qualsevol tipus que sigui. Si la incidència va acompanyada de més d'un fitxer, es recomana utilitzar un algorisme de compressió i empaquetament com ara ZIP o bé RAR.

## VII.- GUIA D'ÚS

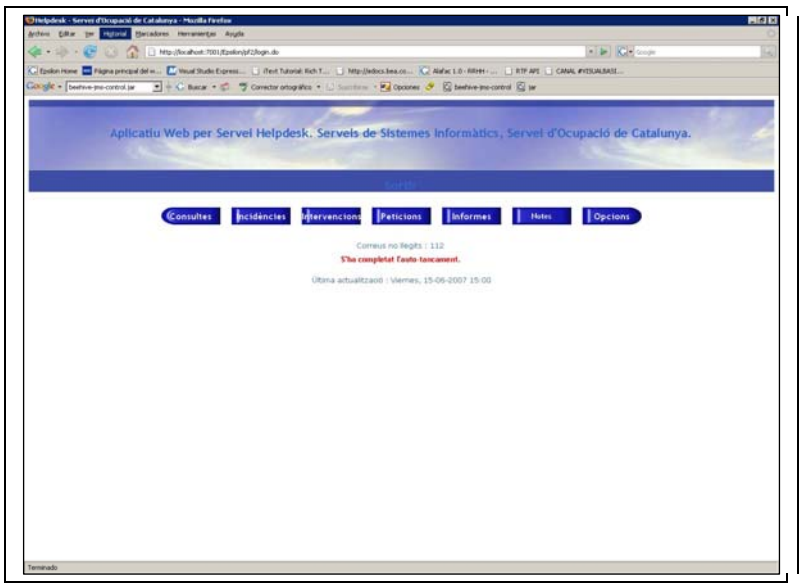
L'aplicatiu té com a punt de partida la pàgina de benvinguda.




En aquesta pàgina, se'ns demana el nom i Password per autentificar-nos al sistema i permetre interactuar amb el mateix.

Un cop autentificats, som conduïts a la pàgina de benvinguda. Des d'aquí accedim al menú principal i se'ns informa del nombre de correus rebuts i no llegits. Des d'aquí podem accedir al sistema de correu fent clic al nombre de correus rebuts.

Si l'aplicatiu s'inicia amb normalitat, veurem un text en color vermell que ens indicarà que s'han tancat les consultes obertes i el nombre de correus rebuts en color blau. En cas de que no sigui possible, degut a un problema amb la base de dades, s'indicaria dins del mateix text. Per l'altra banda, si el sistema de correu no pogués funcionar, el nombre de correus rebuts seria igual a -1.

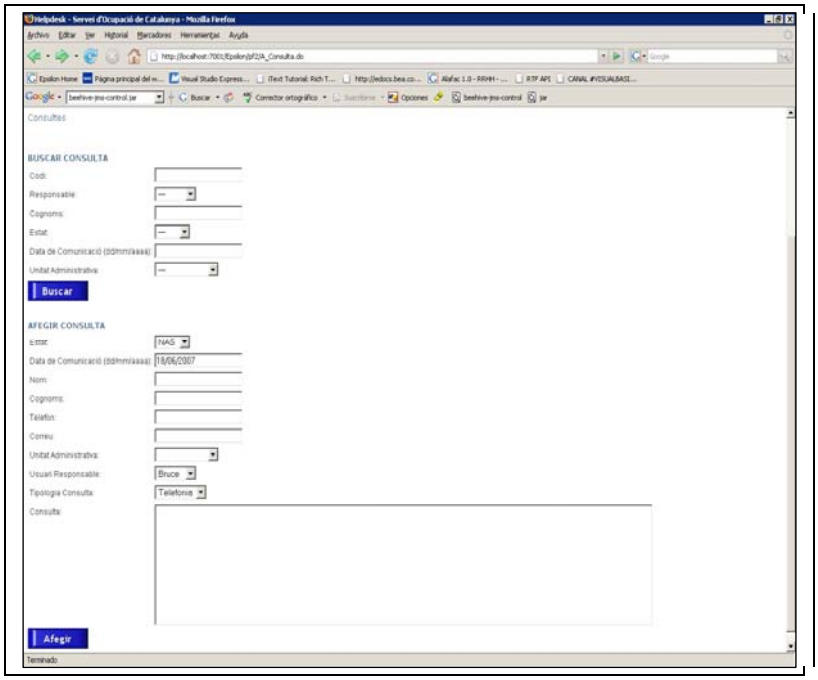



El menú principal consta de les següents opcions :

- Consultes
- Incidències
- Peticions
- Intervencions
- Informes
- Opcions

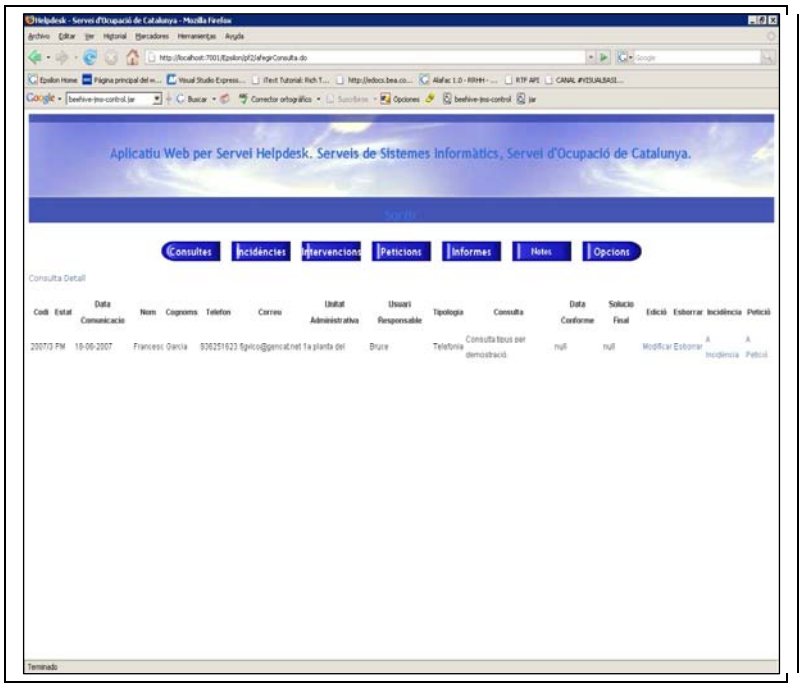
Analitzarem la primera opció d'elles : Consultes

# CONSULTES



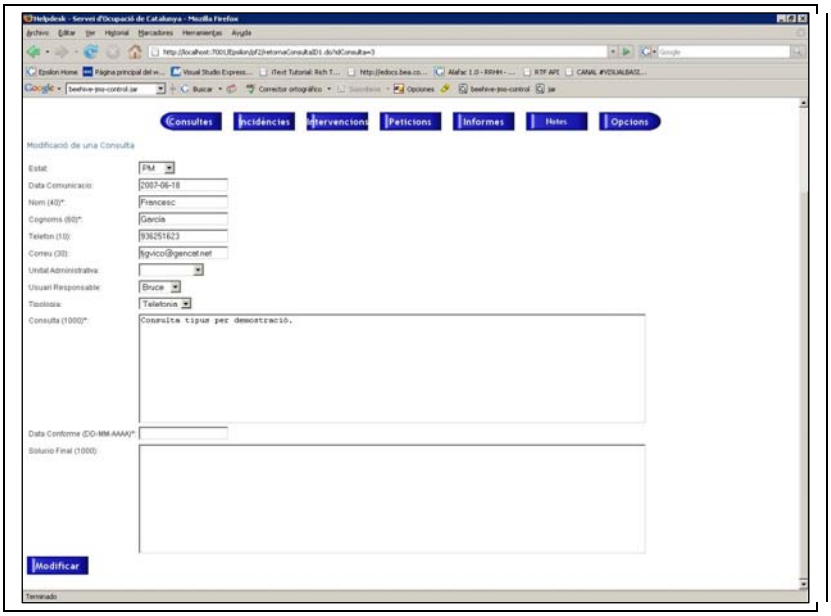

La pàgina de consultes es separa en dues parts: una conté els formularis que permeten accedir a consultes ja guardades. Els criteris per buscar la consulta són : Codi, Usuari Responsable, Cognoms, Estat, Data de Comunicació i Unitat Administrativa.

Si optem per buscar una consulta, introduirem les dades, i l'aplicatiu ens retornarà les consultes coincidents. D'altra manera, podem anar a una altra secció de l'aplicatiu mitjançant les opcions del menú principal, o bé afegir una consulta amb el formulari. Si les introduïm correctament, tenint el compte el nombre de lletres i la codificació de les dates, obtindrem la mateixa pàgina que en el cas anterior, tot i que sols apareixerà la consulta que acabem de crear.



Des d'aquesta pàgina de resultats podem accedir a la consulta mitjançat la opció Modificar i podem eliminar la consulta a través de un formulari de confirmació. Primer provarem de modificar-la.

**Modificar Consulta**



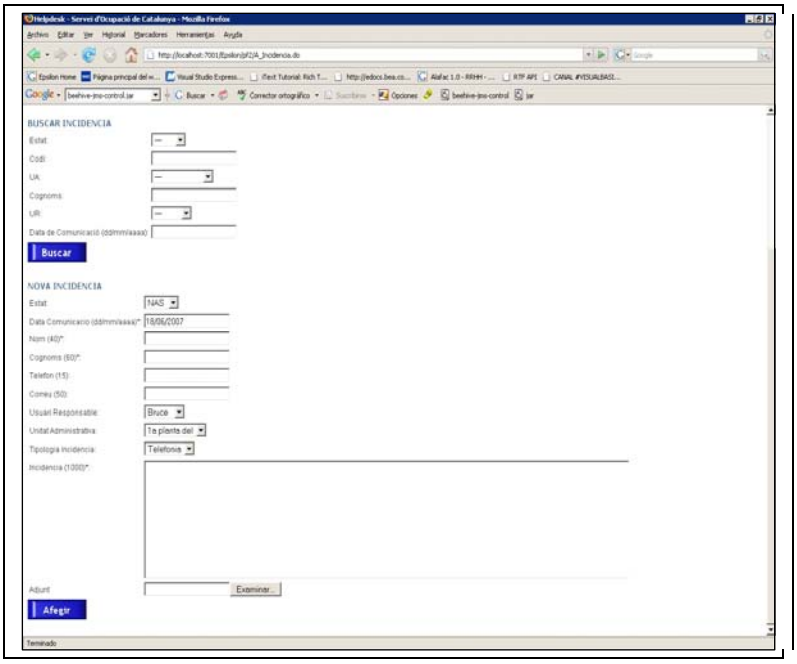

Aquesta pàgina ens retorna un formulari amb la totalitat de les dades de la Consulta. Aquest formulari és el que ens permetrà donar com a tancada una consulta oberta. Un cop actualitzats els camps de la consulta farem clic a Modificar.

**Eliminar Consulta**

Si, en canvi, hem d'esborrar la consulta per qualsevol motiu, mitjançant aquest senzill formulari, se'ns demana la confirmació per eliminar la consulta.

## INCIDÈNCIES

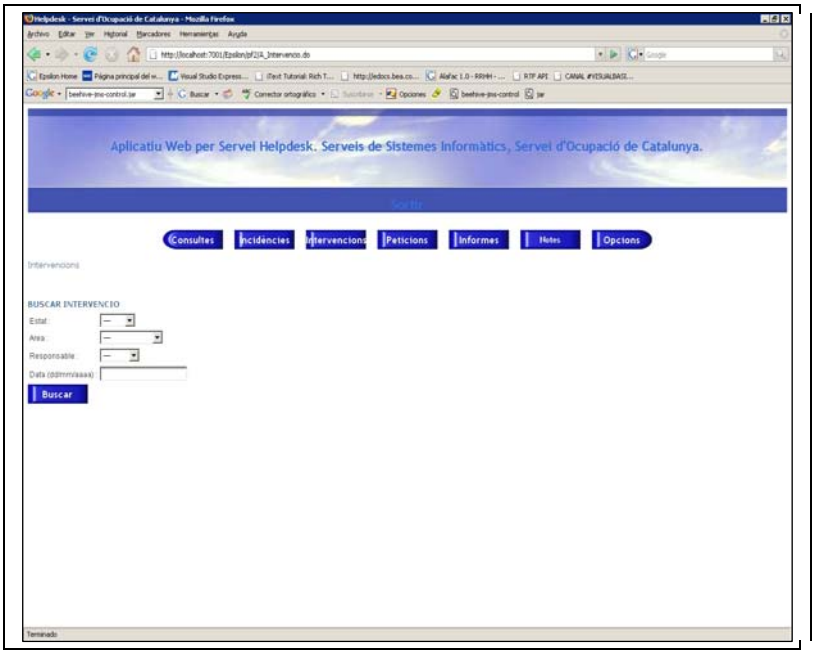
La pàgina d'Incidències també està dividida en una part de cerca, i amb una altra de introducció. En la part de cerca, els criteris són Estat, Codi, Unitat Administrativa, Cognoms, Usuari Responsable i Data de Comunicació. En l'altra part, el formulari d'introducció de Incidències que podem veure en la captura de pantalla :



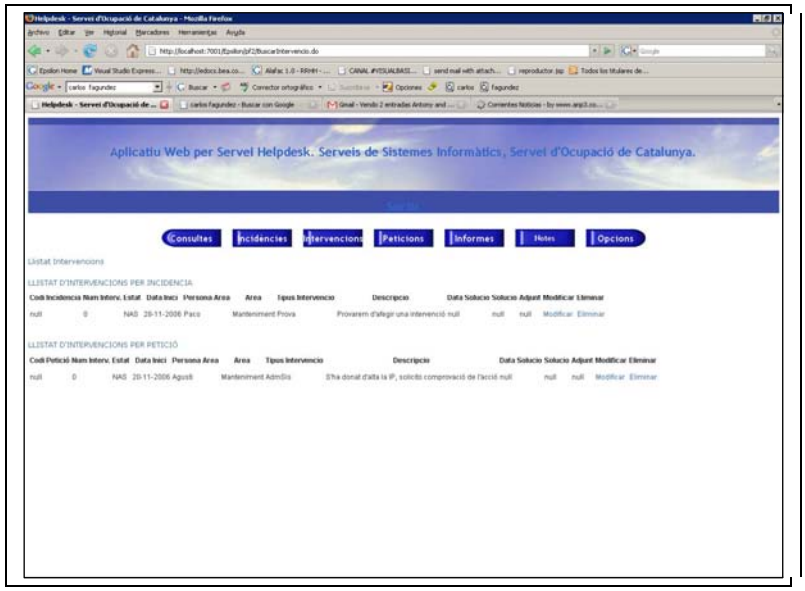

Com en la pàgina de Consultes, aquí també tenim la possibilitat de agregar les incidències directament. Tant la cerca, com la agregació de incidències, ens porten directament a la pàgina de resultats de incidència. Aquesta pàgina és bàsicament un llistat on podem actuar sobre les incidències retornades. Podem, tal com hem fet amb la consulta, modificar-les, a través d'un nou formulari, o bé esborrar-les completament, prèvia confirmació. La tercera opció del menú principal és la següent : Intervencions.

## INTERVENCIONS

La pàgina d'Intervencions sols ens permet cercar Intervencions, indiferentment del origen. Els paràmetres de cerca que disposem són : Estat, Àrea, Responsable i Data. Introduïm un criteri i fem clic a Buscar.

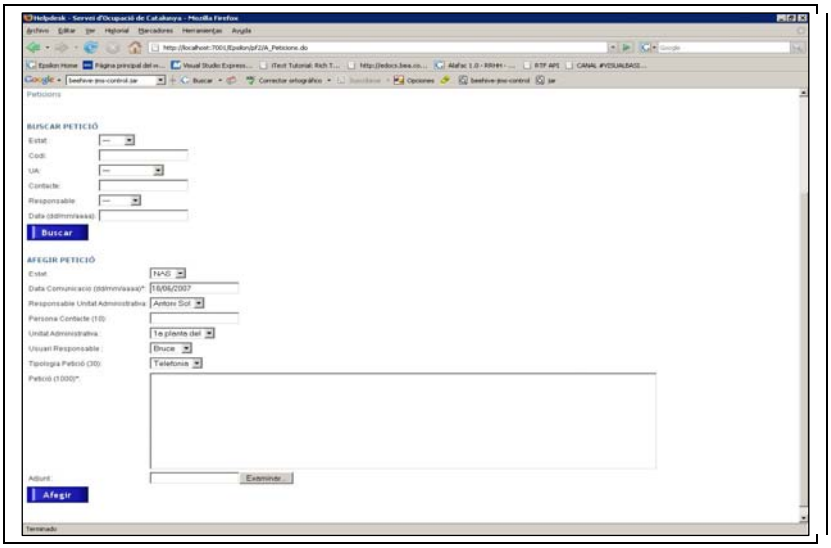



Un cop localitzada/es les intervencions, aquestes són llistades en la següent pàgina Web. Separada per Intervencions d'incidència i Intervencions de petició, es possible de modificar-les o esborrar-les.



# PETICIONS

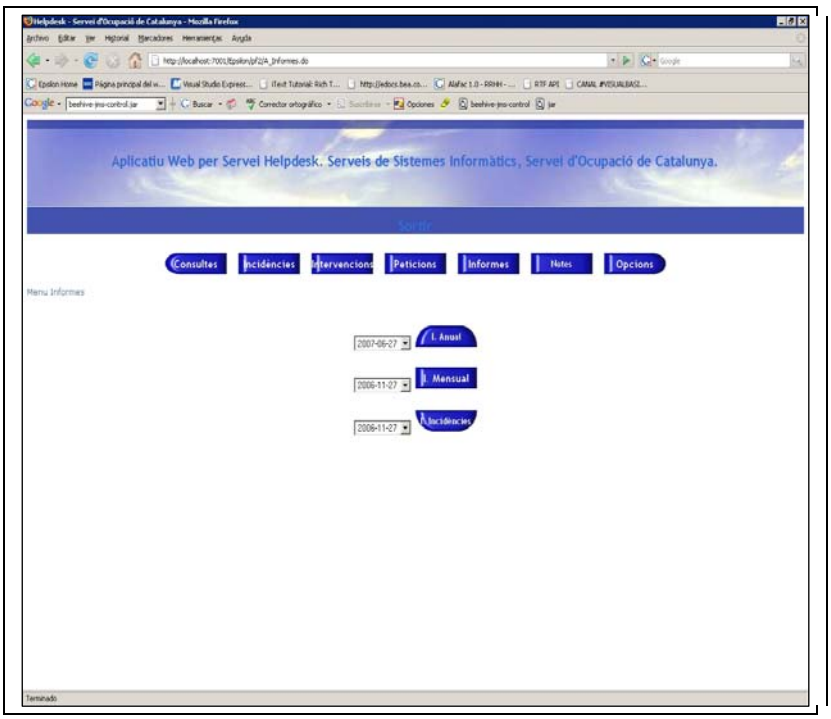
La pàgina de peticions segueix el patró observat a la secció de incidències.



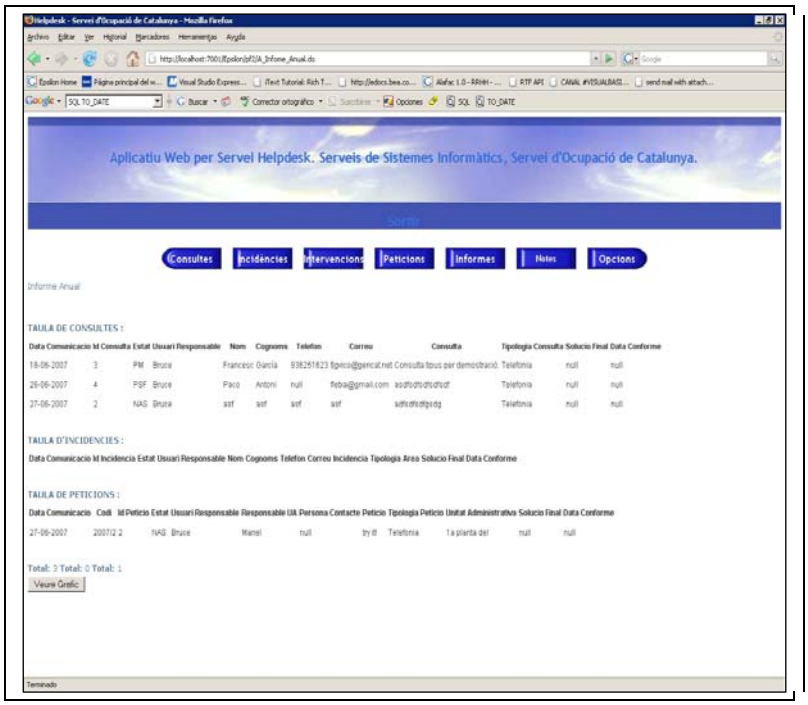
# INFORMES

La pàgina d'Informes, ens permet accedir als mateixos en funció del període i de les dades. Podem escollir entre Informe Anual Complet, Informe Mensual Complet i Informe Mensual d'Incidències i Intervencions.

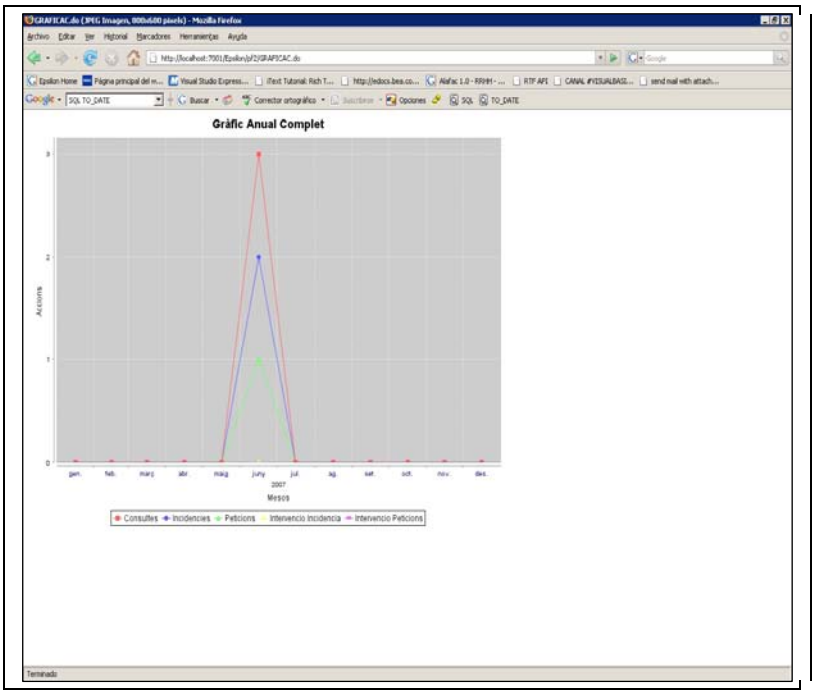
A l'esquerra dels botons, podem observar les respectives llistes desplegables que ens permeten accedir al informes de cadascun del períodes. Per veure els períodes disponibles farem clic al botó petit a la dreta de la data, i es desplegarà el llistat de períodes. Seleccionem el període, i seguidament fem clic al botó Informe per passar a la següent pàgina.



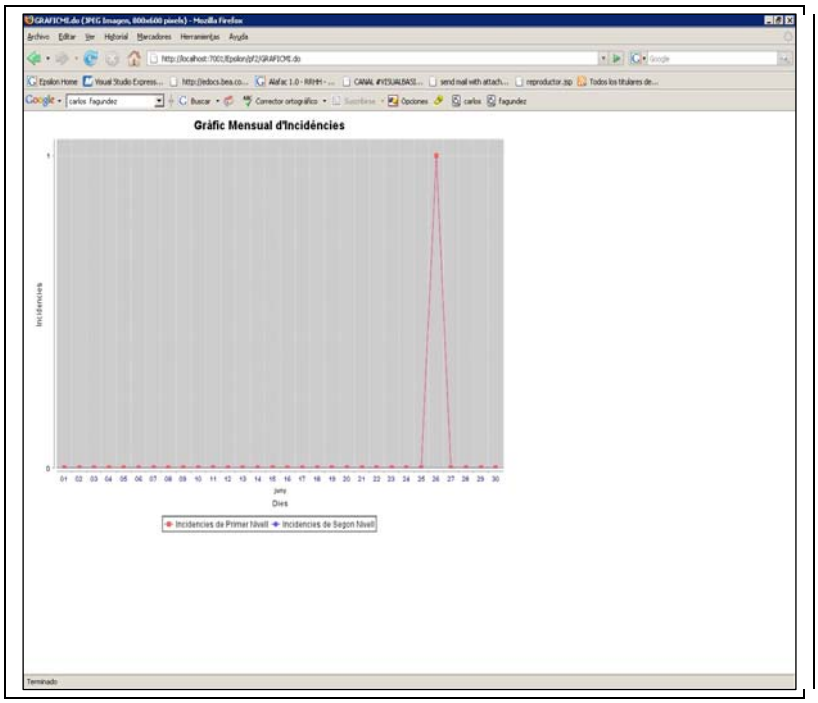

Quan es carregui la pàgina, podrem veure un llistat similar al que apareix en la captura. Podem veure que els llistats es separen per conceptes, si es un informe complet, estarà separat per consultes, incidències i peticions. Si és d'incidències, es separa per incidències de 1r i de 2n nivell. A la part inferior, podem veure els sumatoris de les seccions, i un botó que crida al generador de gràfics.




Podem veure un exemple de representació gràfica en un diagrama de línies.



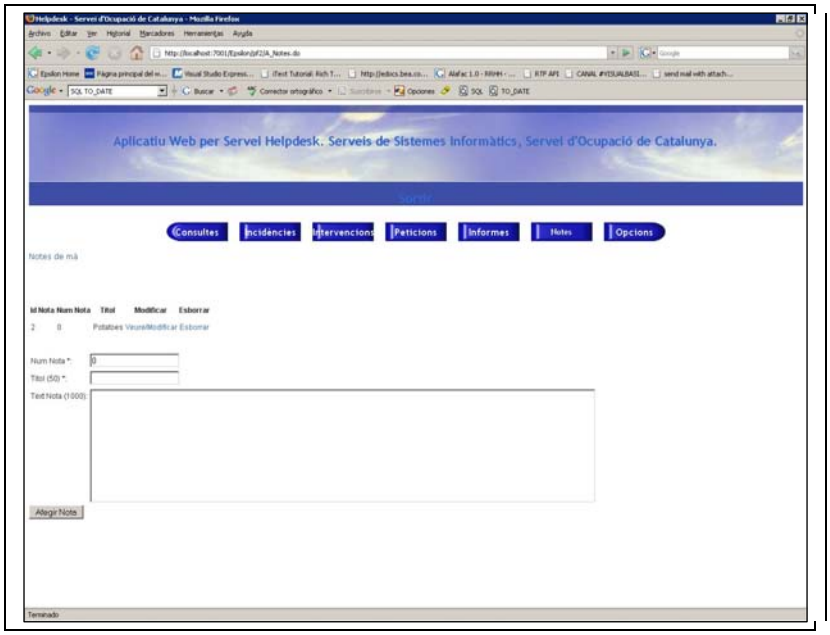

En la captura següent podem veure la representació gràfica d'un Informe Anual Complet. En aquest exemple podem veure com s'han produït tres consultes, dues incidències i una petició durant el mes de juny. Podem observar el interval de temps, separat per mesos a les ordenades, i el nombre de repeticions a les abscisses.




En aquest altre gràfic, d'incidències mensuals, podem veure la divisió del temps en dies naturals del mes de juny a l'eix de les abscisses.

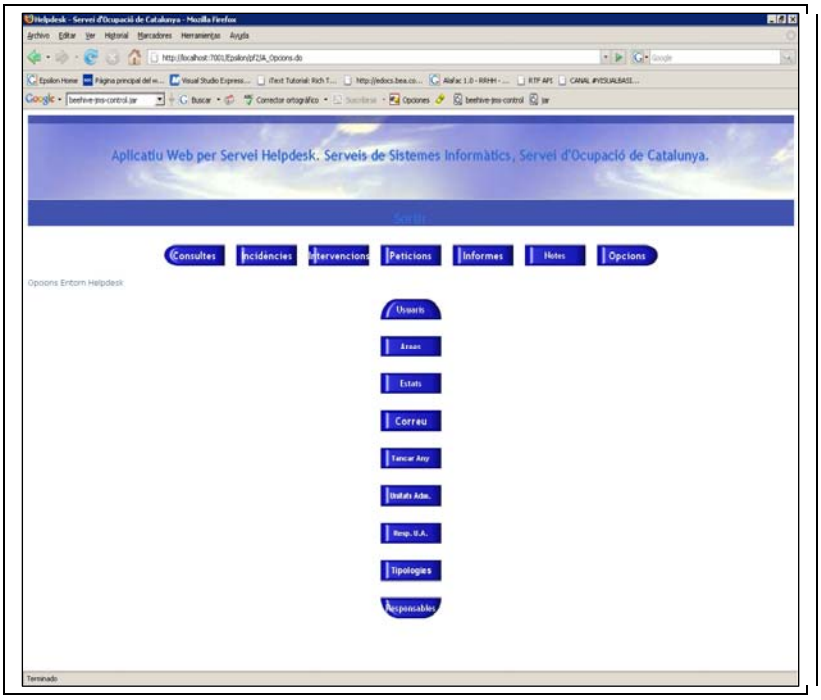
## NOTES

Si seleccionem l'opció del menú principal Notes, accedirem al llistat de notes. En aquest podem accedir al cos de la nota, i també podem esborrar la nota, sense confirmació. A sota el llistat podem veure el formulari per agregar una nova nota. Si seleccionem veure una nota, aquesta ens apareixerà dintre aquest formulari. Per guardar els canvis, farem clic a Afegir Nota, si en canvi, volguéssim sortir sense modificar, podem fer clic sobre una opció del menú principal.

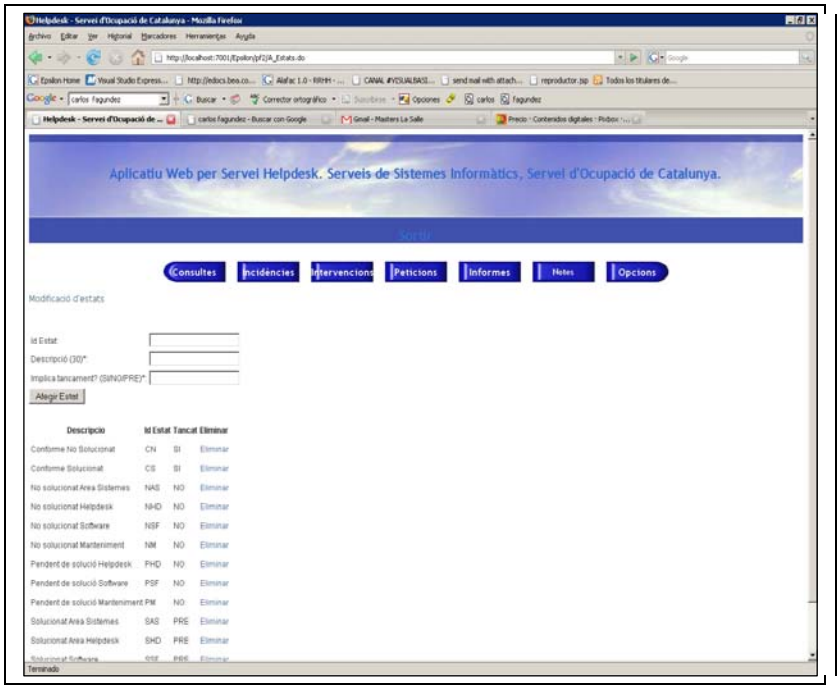


## OPCIONS

Per entrar a la secció d'opcions és necessari estar autenticat com un usuari Administrador, si en canvi, tenim un usuari de perfil operador, no tindrem accés al contingut de la secció d'Opcions. Dintre del menú d'opcions podem seleccionar les opcions que s'han vist en l'apartat V. Mitjançant el menú central, podrem anar als formularis de les diferents opcions, i establir els valors corresponents.



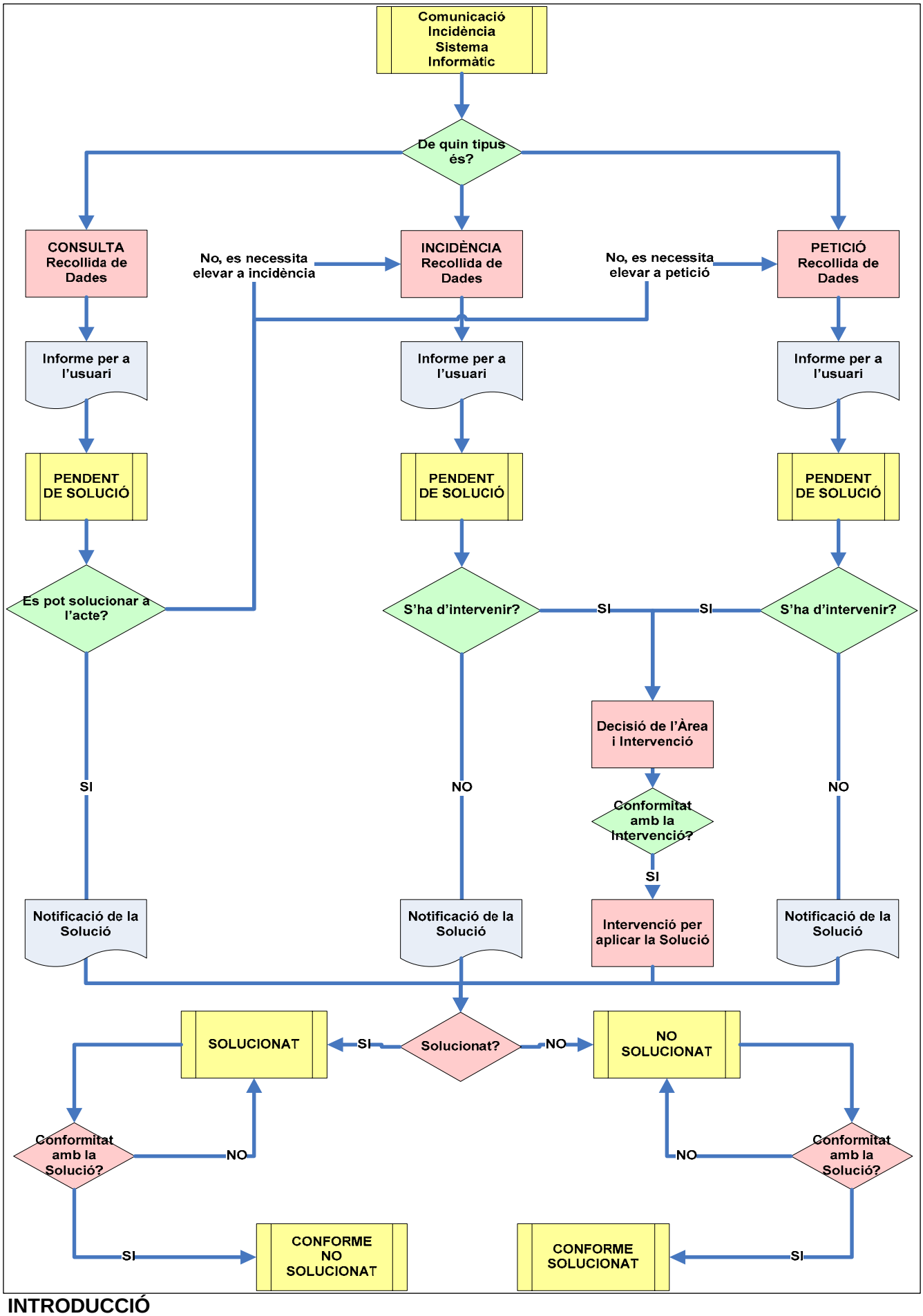

Podem veure, per exemple, el formulari d'opcions dels estats, faríem clic al botó de la secció Opcions, i ens apareixeria la següent pàgina :



Si volguéssim afegir un estat, emplenaríem el formulari de la part superior de la pàgina i faríem clic en afegir. Al completar l'operació veurem un missatge de control que ens indicarà si hem tingut èxit, alhora, podrem veure el nou estat al llistat de la part inferior. Si en cas contrari, volguéssim esborrar un estat, només tenim que fer clic al link al costat de cada element del llistat.

# 11.5 Manual del Programador

**ÍNDEX**




## INTRODUCCIÓ

Aquest software està destinat a la gestió i control de Incidències en la xarxa informàtica del client corporatiu, a través d'una aplicació Web. Substitueix un aplicatiu anterior, afegint noves funcionalitats, i partint d'una base de sistema client - servidor que permet l'accés multi usuari. Aquest software s'ha desenvolupat per al següents servidors :

- Servidor Weblogic 9.2 (Servidor d'aplicacions i J2EE)
- Servidor de BBDD Oracle 8.1.

Per a poder córrer aquesta aplicació es recomana instal·lar-la sobre un PC amb les següents especificacions :

- x86 a 2 GHz
- 2 GB RAM
- 4 GB espai en disc dur.

Està desenvolupat sobre el IDE BEA Workshop for Weblogic 9.2. La gestió de base de dades es realitza mitjançant el driver JDBC que connecta amb una base de dades Oracle. Són recomanables coneixements del llenguatge Java, SQL, Oracle DBA, HTML i CSS.

L'arquitectura d'aquesta plataforma es basa en la següent jerarquia, relacionada estretament amb la pròpia del IDE. Per al desenvolupament de projectes s'ha de crear un workspace específic per al projecte. Un cop creat el projecte. En aquest cas, l'aplicatiu es va crear com un projecte de Pàgina Web Dinàmica o "Dynamic Web Project", unida al que s'anomena un projecte EAR. Podem incloure el projecte EAR al moment de crear el projecte principal. Aquest EAR simplement proporciona una vinculació a l'entorn J2EE. Aquesta aplicació, està dividida en les següents parts :

- Controlador de Flux (.\src\pf2\pf2Controller.java)
- Conjunt de pàgines JSP (.\WebContent\pf2\)
- Conjunt de classes Java (.\src\)
- JAR de JFreeChart. (.\src\servlet\)

## ESTRUCTURA

La gestió de les incidències de la xarxa informàtica es realitza a través d'un protocol d'actuació. Aquest protocol es basa en un servei de Helpdesk que gestiona les consultes efectuades pels usuaris de la xarxa informàtica.

Les incidències es comuniquen amb el servei a través del correu electrònic o bé a través del servei telefònic. Les operadores, segons la consulta, la classifiquen en funció del caràcter i del usuari que realitza la consulta. Aquesta classificació separa aquestes incidències en : Consultes, Incidències de 1r nivell, Incidències de 2n nivell i Peticions.

Les consultes són incidències que poden ser resoltes immediatament a través del correu o bé del telèfon. Per tant, són les de menor importància. A la vegada, són les que necessiten menys dades per tramitar-les.

Les incidències de 1r nivell són aquelles que, tot i que, no poden ser resoltes immediatament a través dels medis preestablerts, són aquelles que poden ser resoltes sense la intervenció de cap tècnic del Servei d'Informàtica.

En canvi, les incidències de 2n nivell impliquen una intervenció d'algun departament del Servei d'Informàtica.

Finalment, les peticions tenen un circuit idèntic al de les incidències, tot i que únicament poden generar-les els caps de Unitat Administrativa, en representació de la seva unitat.

Un cop enregistrades les dades, s'envia un email cap a l'usuari afectat, amb la indicació del codi de la seva consulta. Un cop enviada a confirmació, s'inicia el procés de resolució.

Un cop efectuades aquestes intervencions sobre el sistema, ja bé sigui a través del servei de helpdesk o bé, a través d'una empresa externa, es comunica per part de l'àrea la intervenció que s'ha efectuat l'actuació. Això situa la consulta en un estat d'espera de confirmació. Un cop "solucionada o no" per part de l'àrea, l'usuari ha de ratificar la conformitat o no de la intervenció. Quan s'ha enviat la resposta per part de l'afectat es considera la consulta tancada. Un altre motiu és que es mantingui inactiva durant un mes. Un cop passat aquest termini, la consulta es tancarà amb l'estat "No Conforme". La funcionalitat d'aquest aplicatiu es pot veure amb el següent diagrama :

Per a implementar aquesta lògica en l'aplicatiu s'han desplegat un conjunt de pàgines web JSP, amb les seves accions associades. Aquestes es codifiquen dintre un fitxer anomenat Controlador de Flux, CF. Les accions contingudes pel CF són les següents, seguint el següent ordre jeràrquic :

- Inici
    - Consultes
        - Buscar Consultes per Estat
        - Buscar Consultes per Codi
        - Buscar per Unitat Administrativa
        - Buscar per Cognoms
        - Buscar per Usuari Responsable
        - Buscar per Data de Comunicació
            - Resultat de Buscar
                - Modificar
                    - Guardar
                - Esborrar
                    - Confirmar
                - Passar a Incidència
                - Passar a Petició
        - Afegir Consulta
            - Resultat de l'afegit
    - Incidències
        - Buscar Incidències per Estat
        - Buscar Incidències per Codi
        - Buscar Incidències per Unitat Administrativa
        - Buscar Incidències per Usuari Responsable
        - Buscar Incidències per Data de Comunicació
            - Resultat de Buscar
                - Modificar
                    - Guardar
                    - Afegir Intervenció
                    - Veure Intervencions
                        - Resultat Intervencions
                            - Esborrar Intervenció
        - Afegir Incidència
            - Resultat de l'afegit
    - Intervencions
        - Buscar Intervencions per Estat
        - Buscar Intervencions per Area
        - Buscar Intervencions per Responsable
        - Buscar Intervencions per Data
    - Peticions
        - Buscar Peticions per Estat
        - Buscar Peticions per Codi
        - Buscar Peticions per U. Administrativa
        - Buscar Peticions per Contacte
        - Buscar Peticions per Usuari Responsable

        - Buscar Peticions per Data de Comunicació
        - Afegir Peticions

- Informes
    - Veure Informe Anual Complet
        - Veure Gràfic Anual Complet
    - Veure Informe Mensual Incidències
        - Veure Gràfic Mensual Incidències
    - Veure Informe Mensual Complet
        - Veure Gràfic Mensual Complet

- Notes
    - Afegir Nota
    - Veure Nota

- Opcions
    - Usuaris
    - Àrees
    - Estats
    - Correu
    - Tancar Any
    - Unitats Administratives
    - Responsables Unitats Administratives
    - Tipologies
    - Responsable (Intervencions)

## ELEMENTS

La implementació dels objectes la separem en classes: codificació Java, i taules: modelat BBDD.

**Classe Usuari** Per als usuaris s'utilitza una classe anomenada Usuari, dins del package Model, a més de la corresponent estructura de definició, implementació i connexió a la base de dades. Les propietats d'aquesta classe és la següent :

*private int idUsuari* : Amb aquesta propietat s'indexa l'usuari mitjançant una seqüència numèrica.

*private String Nom* : Aquesta propietat es guarda el nom d'usuari. S'utilitza en l'autenticació i en la identificació del responsable de les incidències.

*private int TipusUsuari* : El nombre TipusUsuari permet distingir entre operadors i administradors. Si el valor d'aquesta propietat és inferior a 500, identifiquem l'usuari com a operador, en cas contrari, s'identifica com a administrador.

*private String Password* : Password s'utilitza per emmagatzemar la contrasenya emprada per autenticar-se al sistema.

*private String Correu* : Aquesta cadena té com a finalitat donar el remitent dels e-mails enviats com a comunicació del informe de sol·licitud.

I els mètodes que s'utilitzen en aquesta secció són els següents :

*public boolean afegirUsuari(Usuari user)* : Mètode per introduir un nou usuari a la base de dades. Aquest nou usuari es crea mitjançant un nou formulari. Retorna un booleà en funció de l'èxit de la operació.

*public Usuari retornaUsuari (int idUsuari)* : Retorna un usuari que tingui el valor de la ID igual al passat per paràmetre. Si no el troba, retorna null.

*public Usuari retornaUsuari(String nom)* : Mètode que retorna un usuari a partir del nom. Si no es troba coincidència, retorna null.

*public String[] retornaNoms()* : Retorna una matriu de cadenes de text amb els noms de cada usuari. Ens permet utilitzar-los en formularis, com a combobox.

*public Usuari[] retornaUsuaris()* : Retorna tots els usuaris entrats a la base de dades per llistar totes les propietats de l'usuari.

*public boolean modificaUsuari (Usuari user)* : Mètode per modificar alguna o totes les propietats del Usuari. S'identifica a través del valor ID del Usuari, i retorna true si es completa l'operació amb èxit.

*public boolean esborraUsuari*(int idUsuari) : Donat un valor numèric, aquest mètode esborra l'usuari segons la ID passada per paràmetre.

*public boolean comprovaUsuari(String nom, String password)* : Aquest mètode comprova que l'usuari existeixi, i correspongui amb el nom i password donats. S'utilitza en l'autenticació a l'hora d'executar accions.

*public Usuari login(String nom, String password) :* Aquest mètode consulta a la base de dades si existeix l'usuari amb el nom proporcionat com a paràmetre *nom*, si existeix, el compara amb el *password* de l'usuari retornat per la base de dades. Si coincideix, retorna la classe Usuari amb les propietats de l'usuari actiu. Si no coincideix, retorna un *null* que és capturat pel controlador de flux com a una autenticació errònia.

**Classe Consulta :** Conté les propietats i mètodes per implementar la lògica definida per a consultes de resolució ràpida. Aquesta classe conté les següents propietats :

*Private int idConsulta* : idConsulta és un camp autonumèric que identifica les Consultes dintre la base de dades.

*Private String CodiConsulta* : Aquesta cadena s'utilitza per a la identificació de la Consulta per part dels usuaris de la xarxa informàtica i dels operadors del Servei de Helpdesk.

*Private String Nom* : Nom del usuari afectat.

*Private String Cognoms* : Cognoms de l'usuari afectat.

*Private String UnitatAdministrativa* : Unitat orgànica a la qual pertany.

*Private String Correu* : Correu electrònic de contacte amb l'usuari

*Private String Telefon* : Telèfon de contacte.

*Private java.util.Date DataComunicacio* : Data de ingrés de la Consulta al Servei de Helpdesk.

*Private String Tipologia* : Tipologia de la consulta.

*Private String Consulta* : Text amb la descripció de la consulta realitzada.

*Private String SolucioFinal* : Text de la Solució que s'ha proposat per resoldre la Consulta.

*Private String Estat* : Estat en que es troba la Consulta, dins la classificació d'estats proposada.

*Private String DataConforme* : Data de Conformitat per part de l'usuari. Aquesta data correspon amb la data en que es canvia a l'estat CN o CS. Es pot donar el cas en que la consulta es tanqui per haver passat un mes natural des de la comunicació de la Consulta.

*Private String UsuariResponsable :* Nom de l'usuari que ha atès la Consulta.

Els mètodes que s'utilitzen dintre la secció Consulta són :

*public boolean afegirConsulta(Consulta consulta)* : Aquest mètode s'utilitza per afegir a la base de dades la consulta que s'ha introduït a través del formulari que es disposa a la primera pàgina de la secció Consultes.

*public Consulta retornaConsultaID(int numconsulta)* : S'utilitza per retornar una consulta en funció del seu número de ID, sobretot en casos de selecció de Consulta dintre un llistat representat en una pàgina JSP.

*public Consulta[] BuscadorConsultes(String Codi, Date Data, String Responsable, String Estat, String Cognoms, String UA)* : Mètode de cerca que permet buscar una o diverses consultes en la base de dades, en funció dels criteris establerts com a paràmetres.

*public Consulta[] RetornaInforme(Date data)* : Retorna un conjunt de consultes que s'han introduït a la base de dades a partir d'una data fins al moment actual. S'utilitza en la secció d'Informes.

*public Consulta[] RetornaInformeDoble(Date data1, Date data2)* : Aquest altre mètode retorna les consultes guardades entre dues dates diferents. La cerca es realitza en funció de les Dates de comunicació

*public Date RetornaPrimeraData()* : Busca en la base de dades la consulta amb la data de comunicació més antiga, de manera que podem indexar el informe en funció de la incidència més antiga.

*public boolean TancaConsultes(Date data)* : Aquest mètode s'utilitza just després d'autenticar-se un usuari dintre l'aplicatiu. Serveix per assegurar que totes les consultes amb un mes natural d'inactivitat es tanquin. Exactament el que fa es consultar l'estat: si està obert el passarà a tancat si entre la data actual i la data de comunicació ha passat un mes natural (30 dies). Retorna un true si no hi ha hagut cap problema (independentment del nombre de consultes tancades).

*public boolean modificaConsulta(Consulta cons)* : Mètode que s'utilitza per a modificar una consulta. S'introdueixen de nou tots els camps, amb l'excepció dels índexs : ID i Codi. Retorna true si s'ha modificat sense errors i false si hi hagut una fallida en el procés.

*public boolean esborraConsulta(int num)* : Definit per eliminar una consulta de la base de dades. A més, l'índex que s'utilitza és la ID. Retorna un true s'hi s'efectua amb èxit.

**Classe Incidències :** Classe que caracteritza tant les incidències de 1r nivell com les de 2n. Comparteix base amb Consulta.

*private int idIncidencia* : Propietat numèrica que identifica la incidència.

*private String Codiincidencia* : Propietat alfanumèrica que identifica la incidència tant per als usuaris com per al Servei de Helpdesk.

*private String nom* : Nom de la persona que ha transmès la incidència.

*private String cognoms* : Cognoms de la persona que ha transmès la incidència.

*private String unitatAdministrativa* : Unitat Orgànica a la qual pertany la persona que ha comunicat la incidència.

*private String correu* : Adreça de correu electrònic de contacte amb l'usuari.

*private String telefon* : Telèfon de contacte amb la persona afectada.

*private java.util.Date dataComunicacio* : Data en què s'ha comunicat la incidència.

*private String Tipologia* : Classificació de la incidència.

*private String incidencia* : Descripció de la sol·licitud.

*private String solucioFinal* : Solució que es dona a la sol·licitud.

*private String estat* : Estat en la que es troba la incidència, segons la classificació.

*private java.util.Date dataConforme* : Data en que o bé s'ha comunicat la conformitat o bé s'ha tancat la incidència de manera automàtica.

*private String usuariResponsable* : Operador del Servei Helpdesk responsable de la incidència.

*private int numIntervencio* : Nombre d'intervencions que s'han fet efectives.

*private String Adjunt* : Camí fins els arxius adjunts de la incidència.

Els mètodes que s'implementen són :

*public boolean afegirIncidencia(Incidencia insi)* : Amb aquest mètode s'agrega la incidència a la base de dades, i retorna un valor true si s'ha realitzat correctament.

*public Incidencia retornaIncidencia(int id)* : Retorna la incidència a partir del valor ID, i null si no s'ha trobat cap coincidència.

*public Incidencia[] BuscadorIncidencies(String Codi, Date Data, String Estat, String UR, String Cognoms, String UA)* : Amb aquest mètode es fa una cerca en funció d'un dels criteris que es passen per paràmetre. Es retorna una matriu d'incidències, o bé null si no s'ha trobat cap coincidència.

*public Incidencia passaIncidencia(Consulta cons,String Area)* : Mètode que genera una incidència a partir d'una consulta i d'una Area de treball, necessària per generar la incidència correctament. Aquest mètode no agrega la incidència directament a la base de dades, simplement la retorna per a introduir-la posteriorment.

*public Incidencia[] retornaInforme(Date DATA)* : Mètode que retorna les Incidències agregades des del paràmetre de temps DATA fins al moment actual.

*public Incidencia[] retornaInformeDoble(Date data1, Date data2)* : Mètode que retorna les incidències comunicades durant un període de temps entre data1 i data2.

*public Incidencia[] retornaInformePrimer(Date DATA)*, *public Incidencia[] retornaInformeSegon(Date DATA)* : Aquests mètodes retornen les incidències des d'una data fins al moment actual si, són de primer grau (sense intervencions) en el primer mètode o bé si són de segon grau (amb intervencions associades) en el segon.

*public Incidencia[] retornaInformeDoblePrimer(Date data1, Date data2)*, *public Incidencia[] retornaInformeDobleSegon(Date data1, Date data2)* : Mètodes que retornen incidències de primer i segon grau que siguin comunicades dintre un període de temps delimitat entre data1 i data2.

*public boolean TancaIncidencies(Date DATA)* : Aquest mètode tanca totes les incidències amb data anterior a la data donada, amb un estat que sigui susceptible de esser tancat per excés de temps : Solucionat / No solucionat. Retorna un booleà en funció de si s'ha completat el procés, independentment del nombre de incidències tancades.

*public Date RetornaPrimeraData()* : Ens retorna la data més antiga de totes les incidències. Ens permet establir els períodes de temps per classificar els informes.

*public boolean modificaIncidencia(Incidencia insi)* : S'utilitza per modificar una incidència. Se li passa una incidència per paràmetre, n'agafa la ID i actualitza els camps de la base de dades pels de la incidència donada. Retorna true en cas de que el procés no doni errors de formatació de camps.

*public boolean esborraIncidencia(int id)* : Mètode que retorna true si s'esborra la incidència amb ID igual al paràmetre.

**Classe Peticions** : La classe peticions comparteix estructura amb la d'incidència, tot i que varien els camps destinats a guardar la informació de l'usuari afectat, ja que en aquest cas, és el cap de la Unitat.

*private int idPeticio* : Valor numèric per identificar la petició dintre la base de dades.

*private String Codi* : Valor alfanumèric que permet als usuaris identificar la petició.

*private String responsableUA* : String amb el nom del responsable de la Unitat Administrativa que ha tramitat la petició

*private String PersonaContacte* : String amb el nom de la persona de contacte en cas de que sigui una persona diferent del Responsable.

*private String UnitatAdministrativa* : String que conté el nom de la Unitat Administrativa on s'origina la petició.

*private java.util.Date dataComunicacio* : Data de la comunicació de la petició.

*private String Tipologia* : String que conté una descripció del tipus de Petició.

*private String Peticio* : String amb una descripció de la petició.

private String solucioFinal : String amb la descripció de la solució.

*private String estat* : String que descriu l'estat.

*private java.util.Date dataConforme* : Data que es comunica la conformitat amb la solució.

*private String usuariResponsable* : String amb el nom de l'usuari que ha introduït la petició.

*private int numIntervencions* : Valor amb el nombre d'Intervencions efectuades.

*private String Adjunt* : String amb la ruta a la carpeta o fitxer que fa referència la petició.

Mètodes :

*public boolean afegirPeticio(Peticio pet)* : Mètode que s'utilitza per afegir la petició a la base de dades. Retorna true o false depenent de l'èxit de l'operació.

*public Peticio retornaPeticioID(int ID)* : Retorna una petició amb ID igual al valor passat per paràmetre.

*public Peticio[] BuscadorPeticions(String Estat, String Codi, String UA, String Contacte, String Responsable, Date Data)* : Mètode per buscar peticions en funció de un dels criteris que es passen per paràmetre. Retorna una matriu de peticions si troba coincidències o null si no en troba.

*public Peticio[] retornaInforme(Date data)* : Retorna una matriu de Peticions que tinguin la Data de Comunicació més propera a la data actual.

*public Peticio[] retornaInformeDoble(Date data1, Date data2)* : Retorna una matriu de Peticions les quals la seva Data de Comunicació estigui entre les dues dates passades per paràmetre com data1 i data2.

*public boolean modificaPeticio(Peticio pet)* : Mètode per modificar la petició que es passa per paràmetre. N'agafa la ID, la cerca dintre la base de dades i n'actualitza els camps pels de la petició que es passa per paràmetre.

*public boolean esborraPeticio(int ID)* : Esborra la petició que tingui una ID igual a la ID passada per paràmetre. Retorna true si s'ha executat l'acció correctament.

*public boolean TancaPeticions(Date data)* : Aquest mètode passa dels estats Solucionat / No solucionat a Conforme, si passen 30 dies de la Data de Comunicació. Retorna true si no s'han produït errors d'accés a disc.

*public Date RetornaPrimeraData()* : Retorna la data més antiga de totes les peticions enregistrades a la base de dades. Permet establir el períodes per a la realització d'informes.

**Classe Intervencions :** es reutilitza tant per Intervencions generades per una Incidència com per les generades per una petició.

*private int idIntervencio* : Enter utilitzat per a la indexació de les intervencions en la base de dades.

*private String CodiIntervencio* : Cadena que identifica la intervenció per als usuaris.

*private String tipusIntervencio* : Cadena utilitzada per guardar la tipologia de la intervenció.

*private String estat* : Cadena que emmagatzema l'estat de la intervenció.

*private String Area* : Cadena amb l'àrea corresponent a la persona que realitza la intervenció.

*private String personaArea* : Cadena del nom de la persona que efectua la intervenció

*private String Descripcio* : Cadena de caràcters per enregistrar la descripció de la intervenció realitzada.

*private String Solucio* : Cadena per guardar el resultat de la intervenció.

*private java.util.Date dataInici* : Data que s'inicia la intervenció.

*private java.util.Date dataSolucio* : Data que es finalitza la intervenció.

*private int intervencioAnterior* : Apunta a la última intervenció feta.

*private int Id* : Enter que apunta a la incidència o la petició de la qual s'ha generat la intervenció.

*private String Adjunt :* Cadena de caràcters que apunta la ruta cap als fitxers relacionats.

*private String Codi* : Cadena de caràcters que apunta al codi de la incidència o petició relacionada.

Els mètodes definits són:

*public boolean afegirIntervencio_I(Intervencio Inter) :* Mètode emprat per afegir una intervenció a la Taula de IntervencionsIncidencia. Retorna true si s'ha completat l'adició, i false si hi ha hagut cap problema.

*public boolean afegirIntervencio_P(Intervencio Inter) :* El mateix que afegirIntervencio_I amb la Taula de IntervencioPeticio.

*public boolean modificarIntervencio_I(Intervencio Inter), public boolean modificarIntervencio_P(Intervencio Inter) :* S'utilitza per modificar els valors d'una intervenció, per la taula IntervencioIncidencia i IntervencioPeticio. Es passa per paràmetre la intervenció i es retorna true si s'ha completat la modificació

*public Intervencio[] retornaIntervencioId_I(int id), public Intervencio[] retornaIntervencioId_P(int id) :* Mètodes que retornen una matriu de Intervencions segons la ID proporcionada, aquesta correspon a la Incidència o Petició que les originen. Si no es troben coincidències, es retorna null.

*public Intervencio retornaIntervencio_I(int id), public Intervencio retornaIntervencio_P(int id) :* Mètodes que retorna la Intervenció amb la ID proporcionada.

*public Intervencio[] BuscarIntervencioI(Date Data, String Num, String Area, String Estat), public Intervencio[] BuscarIntervencioP(Date Data, String Num, String Area, String Estat)* : Mètodes que permeten cercar Intervencions en funció d'un criteri dels que apareixen com a paràmetre. Es necessita passar un paràmetre com a mínim, si no, es retorna null.

*public boolean esborraIntervencio_I(int id), public boolean esborraIntervencio_P(int id) :* Mètodes que fan la funció d'eliminar de la base de dades les Intervencions amb la id igual al paràmetre passat. Retorna true si es completa la operació.

*public boolean esborraIntervencions_I(int IDIncidencia), public boolean esborraIntervencions_P(int IDPeticio) :* Mètodes que permeten esborrar les intervencions generades per les Incidències o Peticions amb id igual al paràmetre passat. Igual que amb *esborraIntervenció*, retorna true si es completa amb exit.

*public Intervencio[] retornaInformeI(Date data), public Intervencio[] retornaInformeP(Date data):* Mètodes que retornen els informes de les incidències i les peticions entre la data i la data actual.

*public Intervencio[] retornaInformeDobleI(Date data1, Date data2), public Intervencio[] retornaInformeDobleP(Date data1, Date data2):* Retorna les intervencions tant de Incidències com de Peticions entre dues dates.

*public int getAny()* : Retorna l'any fins el qual es consideren totes les incidències tancades. I a partir del qual, les incidències són operatives excepte les que estiguin expressament tancades.

*public boolean setAny(int ANY) :* S'introdueix l'any de tancament. Valor que farà que totes les incidències anteriors a aquesta es consideren tancades.

**Classes OpcionsCorreu.java :** En aquesta classe es guarden les opcions de configuració per al servei de e-mail intern de l'aplicatiu. S'utilitza per a la comunicació interna entre mètodes.

*int ID :* Valor numèric per a la indexació

*private String Nomusuari :* String que conté el nom d'usuari del compte de correu.

*private String Password :* String amb el password del usuari que s'utilitza per l'enviament de correu.

*private String Smtphost :* String amb el nom o la IP del servidor de correu SMTP.

*private int Smtpport :* Port d'accés al servei SMTP del servidor SMTP.

*private String Pophost* : String amb el nom o la IP del servidor POP3 de correu

*private int Popport :* Valor numèric amb el port d'accés al servei POP3.

*private String Cospeticio :* String amb el cos del correu que serà introduït automàticament en els emails que es generen quan s'agrega una petició.

*private String Cosincidencia :* String que conformarà el cos dels emails que es genera automàticament quan s'agrega una incidència.

*private String Cosintervencio :* String que conté el missatge que anirà al cos del email per a informar de la Intervenció.

*private String COSRTF :* String que conté els caràcters que formen l'informe en format RTF que s'adjunta en cada email.

*private String RutaGuardar :* Camí per guardar les dades que es reben del servidor de correu POP.

**Classe Correu :** Aquesta classe implementa les dades necessàries per a la caracterització d'un e-mail, junt amb algunes dades per a l'enviament.

*private String Host :* String que guarda l'adreça del servidor SMTP de correu electrònic.

*private String Remitent :* String on registrar l'adreça del remitent del correu electrònic.

*private String Destinatari :* String on s'emmagatzema l'adreça del destinatari del correu electrònic.

*private String Assumpte :* String on s'inclou el que serà l'assumpte dintre el corre electrònic.

*private String Cos :* String que es correspon amb el cos del missatge del correu electrònic.

*private String* : String corresponent a la CC o Carbon Copy del email.

*private boolean isHTML* : Valor booleà, que indica si el correu s'ha d'enviar en format text (false) o bé HTML (true).

*private boolean debug :* Valor booleà que informa l'aplicatiu que ha de mostrar els missatges de control durant el procés d'enviament a la finestra de terminal del servidor.

*private org.apache.struts.upload.FormFile AdjuntFormFile :* Objecte FormFile, utilitzat per recuperar fitxers del PC del client, per tal de poder-lo adjuntar amb el correu electrònic.

*private String Adjunt :* Cadena de caràcters que representa la informació que s'adjuntarà amb el correu, en aquest cas, correspon al cos del fitxer de text en format RTF.

*private int index :* Valor numèric que permet identificar el correu electrònic dintre la matriu

*private Date data :* Valor Date que enregistra la data d'enviament del correu electrònic.

*private MimeBodyPart RTF :* part del missatge de correu electrònic que es destina exclusivament al informe RTF que s'inclou en les altes de Incidències i Peticions.

*public boolean send(Correu C) :* Mètode que implementa l'enviament de correus electrònics mitjançant la classe Correu. Retorna false si hi ha hagut algun error en l'enviament, i true si s'ha completat el procés.

*public boolean comunicaIncidencia(Incidencia I) :* Mètode que implementa l'enviament automàtic del informe a través e-mail a l'usuari afectat. Es passa per paràmetre la incidència per tal d'actualitzar les dades de l'informe en format RTF que s'adjunta.

*public boolean comunicaPeticio(Peticio P) :* Al igual que en comunicaIncidencia, implementa l'enviament automàtic d'un correu electrònic amb el informe de la petició adjunt.

*public Correu[] rebre() throws MessagingException, IOException :* Mètode que recull els missatges rebuts al compte, que es guarda a la base de dades, i els emmagatzema en un objecte anomenat mailbox. Els adjunts dels correus es guarden en la ruta introduïda a les opcions de correu.

*public Correu[] rebreAssumptes() throws MessagingException, IOException :* Permet rebre només els assumptes dels correus rebuts, així s'estalvia temps de descàrrega.

*public String getAssumpte(int idUsuari, int idPrincipal, int idSecundari, String text, boolean esPeticio) :* Mètode que construeix un String que serà utilitzat al correu electrònic de informe. Crea una cadena de text resultant de la combinació del codi, l'usuari, la data, etc.

*public int getCorreusNous(OpcionsCorreu oc) throws MessagingException :* Mètode que recull el nombre de correus que tenen la marca de no llegits. Aquest valor numèric es retorna com a resultat de l'execució. Si es produeix un error, retorna -1.

*public int getTemps() :* Mètode per recuperar el nombre de dies per a procedir al tancament automàtic.

*public boolean setTemps(int n) :* Mètode per emmagatzemar el nombre de dies per a executar el tancament automàtic.

*public Correu[] actualitzaFolder() throws MessagingException, IOException :* Mètode que implementa la recepció de correus electrònics.

*public OpcionsCorreu getOpcions() :* Opció per recuperar les opcions del servei de correu emmagatzemades a la base de dades, per ser mostrades en la pàgina web.

*public boolean setOpcions(OpcionsCorreu oc) :* Mètode per guardar les opcions del servei a la base de dades.

*public boolean esborraCorreu(int index) :* Mètode que esborra un missatge de la bústia del servidor de correu, segons l'índex que se li passi per paràmetre.

**Classe Area :** Aquesta classe guarda els valors que defineix l'àrea. Aquests són bàsicament un conjunt de String que permeten caracteritzar l'àrea.

*private int idArea* : Identificador numèric

*private String NomArea* : String que guarda el nom de l'àrea.

*private String Situacio* : String que descriu la localització física.

*private String Telefon* : String que recull el número de telefon.

*private String Correu* : String on es registre l'adreça de correu electrònic.

*private String Director* : String on es guarda el nom de la persona que ocupa la direcció de l'Àrea.

Els mètodes són :

*public boolean NovaArea(Areas concor)* : mètode emprat per a la introducció a la base de dades de noves àrees. Retorna un boolean en funció de l'èxit de la operació.

*public boolean EsborraArea(Areas concor)* : mètode utilitzat per esborrar la àrea de la base de dades. Retorna un boolean en funció de l'èxit de l'operació.

*public String[] RetornaAreas()* : mètode utilitzat per retornar les cadenes de text corresponents al nom de cadascuna de les àrees.

*public Areas[] RetornaAreesTotes()* : mètode utilitzat per retornar totes les àrees, amb la intenció de ser llistades en pàgina. Retorna una matriu de classe Area.

**Classe Estats** : La classe estats recull els estats en que podem classificar una incidència.

*private String IdEstat* : String que representa l'acrònim de l'estat.

*private String Descripcio* : String que guarda el nom complet de l'estat.

*private String Tancat* : String que conté si l'estat implica tancament, no o si es susceptible de ser tancat per superació del l'interval de temps d'obertura.

Els mètodes son :

*public boolean NouEstat(Estats NOU)* : Mètode que afegeix un estat a la base de dades per a la seva utilització en els formularis. Retorna un booleà en funció de l'èxit de l'operació realitzada.

*public boolean EsborrarEstat(String IdEstat)* : Mètode utilitzat per esborrar l'estat. Retorna un booleà que depèn de l'èxit de l'operació.

*public String[] RetornaEstats()* : Mètode que retorna una matriu de cadenes de text amb la ID de cada estat. S'utilitza en formularis.

*public Estats[] RetornaTot()* : Mètode que retorna una matriu de classe Estat, s'utilitza per llistar els estats en pàgina JSP.

*public String[] RetornaEstatsTancats()*: Mètode que retorna una matriu de cadenes de text amb la ID del estats que representen el tancament d'una incidència.

*public String[] RetornaEstatsOberts()*: Mètode que retorna una matriu d'estats que implementen que la incidència està oberta.

**Classe Nota.java :** La classe Nota recull les notes, que seran registrades en la base de dades per ésser compartides entre els usuaris de l'aplicatiu.

*private int idNota:* Valor numèric que indica l'identificador de la nota.

*private String Nota:* String amb el text complet de la nota. Està limitat a 1000 caràcters

*private String Titol :* String per guardar un títol descriptiu de la nota.

*public Nota afegirNota(Nota Nota)* : Mètode per introduir una Nota nova, retorna un true si es completa la introducció.

*public Nota retornaNota(int id)* : Mètode que retorna la nota amb la ID indicat per referència.

*public Nota[] retornaNotes()* : Retorna totes les notes guardades a la base de dades.

*public boolean esborraNota(int idnota)* : Esborra la nota guardada dintre la base de dades, amb ID igual al valor passat per paràmetre.

*public String[] retornaTitol()* : Retorna els títols de les notes guardades en base de dades.

**Classes Responsable.java :** Aquesta classe s'implementa per retenir les dades dels responsables de les intervencions.

*private int idResponsable :* Valor numèric per indexar del tècnic de l'Àrea..

*private String nomResponsable :* String que conté el nom i cognoms del tècnic de l'Àrea.

*private String correuResponsable :* String amb el correu electrònic de la persona responsable.

*private String area :* String amb el nom de la Àrea a la qual pertany.

Mètodes :

*public boolean afegirResponsable(Responsable resp) :* Mètode per introduir un Responsable a la base de dades. Retorna  true si s'ha afegit correctament.

*public String[] retornaNoms() :* Mètode que retorna una matriu de cadenes de text, amb els noms dels tècnics de les àrees disponibles. S'utilitza en formularis.

*public Responsable retornaResponsable(int id) :* retorna el responsable a partir del valor de ID, si no es coincideix retorna null.

*public String retornaCorreuN(String nom) :* Retorna un String contenint el correu del Responsable que es correspon amb el nom passat.

*public Responsable[] retornaTots() :* Retorna una matriu de la classe Responsable amb tots els Responsables existents a la base de dades.

*public boolean esborraResponsable(int id) :* Utilitza la ID per esborrar el Responsable coincident. Si es completa l'operació retorna un true.

**Classe ResponsableUnitat.java :** Amb aquesta classe s'identifica als cap de Unitat Administrativa, per relacionar-los directament amb les peticions.

*private int idResponsable :* Valor numèric utilitzat per indexar els Responsables de la Unitat.

*private String NomResponsable :* String que conté el Nom i Cognoms  de la persona Responsable de la Unitat Administrativa, que s'identifica especialment per fer Peticions.

*private String Correu :* String amb l'adreça electrònica corresponent al Responsable de la Unitat Administrativa.

*private String UA :* String que fa referència al nom de la Unitat Administrativa.

*private String Telefon :* String on es registra el telefon del Responsable.

Mètodes :

*public boolean afegirResponsableUnitat(ResponsableUnitat RU) :* Mètode per introduir un Responsable de la Unitat Administrativa a la base de dades. Retorna un true si s'efectua amb èxit.

*public String[] retornaResponsablesUnitat() :* Mètode que retorna una matriu de Cadenes de text amb els noms dels Responsables, per ser utilitzada en formularis.

*public ResponsableUnitat retornaResponsableUnitat(int ID) :* Retorna un Responsable de la Unitat Administrativa pel valor de ID. Si no coincideix, retorna un null.

*public ResponsableUnitat[] retornaResponsableUnitat() :* Retorna una matriu de classes ResponsableUnitat, amb la totalitat de entrades a la base de dades. S'utilitza per llistar les dades en una sola pàgina.

*public boolean esborrarResponsableUnitat(ResponsableUnitat RU) :*Mètode per esborrar el Responsable de la Unitat Administrativa, mitjançant la classe Responsable que es passa per paràmetre.

**Tipologia.java : La classe tipologia permet crear un sistema d'etiquetes amb el que classificar fàcilment les incidències.**

*private String Nom :* Nom de la Tipologia a mostrar en els formularis.

*private String Descripcio :* Descripció curta de la Tipologia.

Mètodes :

*public boolean afegirTipologia(Tipologia Tip) :* Mètode per introduir la Tipologia a la base de dades. Retorna true en cas d'èxit.

*public String[] retornaTipologies() :* Retorna una matriu de String amb els noms de les Tipologies, amb l'objectiu de ser utilitzats en formularis.

*public Tipologia[] retornaT() :* Retorna una matriu de classes Tipologia, on es registren les Tipologies guardades a la base de dades.

*public boolean esborraTipologia(Tipologia Tip) :* Mètode per esborrar la tipologia que es passi per paràmetre de la base de dades. Si s'esborra correctament, el mètode retorna true.

**UnitatAdministrativa.java :** guarda les dades relatives a les unitats orgàniques de la xarxa informàtica del client.

*private int Id :* Valor numèric utilitzat per indexar les unitats Administratives.

*private String Nom :* String per identificar-la Unitat Administrativa a la que pertany l'usuari afectat.

*private String Direccio :* String que guarda l'adreça o descripció per localitzar de la Unitat Administrativa, en cas que s'hagi de desplaçar una persona per efectuar una Intervenció.

*private String Telefon :* String per enregistrar un telèfon de contacte amb la Unitat Administrativa en general, o amb el seu cap, en particular.

*private String Correu :* String que conté una adreça de correu electrònic per informar l'Àrea, per exemple, d'un avís, o bé una notícia.

*private String Responsable :* String amb el nom i cognoms de la persona que té les funcions de cap o responsable de la Unitat Administrativa.

Mètodes :

*public boolean afegirUnitatAdministrativa(UnitatAdministrativa UA) :* Mètode per introduir la Unitat Administrativa a la base de dades. Retorna un booleà en funció de l'èxit de la operació.

*public String[] retornaUnitatsAdministratives() :* Mètode que retorna una matriu de cadenes de text que són utilitzats en formularis.

*public UnitatAdministrativa[] retornaUAs() :* Mètode que retorna una matriu de objectes UnitatAdministrativa, per llistar totes les propietats en pantalla. S'utilitza en la pàgina "UA.jsp".

*public UnitatAdministrativa retornaUA(int ID) :* Mètode que retorna la Unitat Administrativa que es correspon amb la ID que se li passa per paràmetre. Si no la troba, retorna un null.

*public boolean esborraUnitatAdministrativa(UnitatAdministrativa UA) :* Mètode que esborra la Unitat Administrativa.

---

## ELEMENTS EXTERNS

A més, en el cas d'aquesta aplicació, a l'hora de generar els gràfics, es recorre a una entitat externa, **JFreeChart** :

> **JFreeChart** és un Servlet que ens permet dissenyar gràfics a partir dels paràmetres que li són enviats. S'ha connectat al conjunt mitjançat la incorporació del fitxer JAR dintre del conjunt de llibreries emprades pel servidor. Alhora d'executar-lo, genera un arxiu JPG que es visualitzat a través del navegador Web. Aquesta entitat externa és un Servlet independent del controlador de flux. Per a incorporar-lo, s'afegeix al build path del projecte, i es crea un fitxer Java amb la implementació necessària per a l'ús per part del nostre aplicatiu. Després per cridar-lo, al JSP, en lloc d'apuntar-lo a una acció, s'apunta al Servlet directament. Per a passar-li les dades, s'utilitzen les capçaleres HTTP, on afegirem com a paràmetres les classes a utilitzar. En la banda del Servlet, haurem de fer una conversió de nou cap a la classe original, ja que els paràmetres es retornen com a String.

## 11.6 Procés de Instal·lació de l'aplicatiu

**Requeriments**

Processador x86 > 2,8 GHz
> 1 GB RAM
~ 3,5 GB espai al disc dur

**Procediment**

1. Instal·lar ORACLE 8i :
   Instal·lació típica, Nom BBDD : HELPDESK, SID : HELPDESK

2. Instal·lar PORTAL 920 :
   Opcions per defecte

3. Importació Base de dades :
   1. Importar EXPDATA1.DMP com a usuari : SYSTEM / MANAGER
   2. Importar EXPDATA2.DMP com a usuari : UHELPDESK / HELPDESK

4. Importació projecte WEBLOGIC :
   Creació d'un projecte :
      Dynamic Web Project : Epsilon, amb EAR : EpsilonEAR
      Opcions per defecte

   Creació de Server :
      Creació de Domini :
         Opcions JDK : Development / Sun
         Opcions JDBC : Oracle, oracle, Thin, helpdesk, localhost, 1521, uhelpdesk, helpdesk
         Transfer Type : Cache-flush

5. Copiar JFreeChart i JCommon, esborrar enllaços anteriors i afegir els nous.

6. Afegir nou WorkManager al servidor.

## 11.7 Model de programació

Programació Extrema

- Desenvolupament iteratiu i incremental:petites millores.
- Proves unitàries continues, freqüentment repetides i automatitzades, incloent proves de regressió. S'aconsella escriure el codi de la prova abans de la codificació. Per exemple, les eines de prova JUnit orientada a Java y DUnit orientada a Delphi.
- Programació en parelles: es recomana que les tasques de desenvolupament es duguin a cap per dues persones en un mateix lloc. Es suposa que la major qualitat de codi, escrit d'aquesta manera. El codi es revisat i discutit mentre s'escriu, és més important que la possible pèrdua de productivitat immediata.
- Freqüent interacció del equip de programació amb el client o usuari. Es recomana que un representant del client treballi junt amb l'equip de desenvolupament.
- Correcció de totes les errades abans d'afegir una nova funcionalitat. Fer entregues freqüents.
- Refactorització del codi, és a dir, reescriure certes parts del codi per augmentar la seva legibilitat i mantenibilitat però sense modificar el seu comportament. Les proves han de garantir que en la refactorització no s'ha introduït cap errada.
- Propietat del codi compartida: en lloc de dividir la responsabilitat del desenvolupament de cada mòdul en grups de treball, aquest mètode promou que tot el personal pugui corregir i estendre qualsevol part del projecte. Les proves freqüents de regressió garanteixen que les possibles errades siguin detectades.
- Simplicitat en el codi: és la millor manera de que les coses funcionen. Quan tot funcioni es podrà afegir funcionalitat si és necessari. La programació extrema aposta que és més senzill fer una cosa simple i tenir més feina per canviar-la si és necessari, que no pas fer quelcom complicat i possiblement no utilitzar-ho.

## 11.8 Plec de requeriments de l'aplicatiu del client

Les incidències reportades pels usuaris en relació als sistemes d'informació es gestionen mitjançant una aplicació informàtica (Helpdesk) que requereix unes noves funcionalitats.

L'aplicació ha de permetre gestionar de manera diferenciada les consultes, les incidències i les peticions que arriben al Departament pels diferents canals establerts a aquests efectes (correu electrònic, trucades, ...)

**Consultes :** Registrar les consultes que els usuaris realitzen al servei de Helpdesk i que es solucionen telefònicament. Les dades necessàries han de ser les mínimes, nom i cognoms de la persona que fa la consulta, la seva unitat orgànica, la tipologia de la consulta i solució. Una vegada resolta, es tanca o dóna lloc a una incidència o petició. Si és així, s'ha de permetre mantenir les dades ja introduïdes, excepte en el cas que sigui una petició que requereixi la comunicació per part del cap de la unitat corresponent.

**Incidències :** S'ha de diferenciar entre incidències de primer nivell que són resoltes directament pel servei de Helpdesk i incidències de segons nivell que són redirigides a d'altres àrees.

Incidències de primer nivell: es registra la persona que comunica la incidència (nom i cognoms, unitat orgànica, telèfon, correu electrònic), la descripció de la incidència i la solució aplicada pel servei de Helpdesk. El fet de registrar la incidència ha de generar un correu electrònic que s'enviarà a la persona que l'ha obert on se li comunica el núm. d'incidència assignat i que servirà per a consultes posteriors. El fet de solucionar una incidència suposa un vist i plau de l'usuari que permetrà fer el tancament definitiu.

Incidències de segon nivell: són les que no poden ser resoltes directament pel servei de Helpdesk i són redirigides a les diferents àrees del Departament: Sistemes, Manteniment o Programari.

El procediment per a la gestió d'aquesta tipologia d'incidències és el mateix que en el cas anterior però permetrà assignar-la a una persona de les àrees esmentades i s'enviarà a través del correu electrònic amb els documents annexos que calgui.

**Peticions :** La única diferència entre aquest circuit i l'anterior és que les persones autoritzades per tramitar peticions són els caps de les unitats corresponents.

Tot això fa que una necessitat important en la gestió de les incidències sigui el mòdul de cerques que ha de permetre fer-les en funció de:

Núm. d'incidència/petició
Data d'incidència
Persona Responsable
Àrea
Estat

Així mateix, l'aplicació ha de permetre l'elaboració d'informes estadístics (taules i gràfics) de la tipologia següent:

Informes mensual indicant el total de consultes, incidències i peticions.
Informes mensuals de les incidències diferenciant entre les de 1r i 2n nivell.
Informe acumulatiu anual.

## 11.9 Especificacions de la plataforma

L'aplicació és basa en un servidor d'aplicacions "propietari", desenvolupament a partir de Tomcat, que és un projecte GPL de servidor d'aplicacions Java. Sobre aquest servidor d'aplicacions, anomenat BEA Weblogic s'ha unit Struts, una part que simplifica l'aplicació web creant "fluxos de pàgines". Això estableix uns camins i uns nodes que facilita el plantejament de l'aplicació. Els camins són les accions, i els nodes són les pròpies pàgines web. Les accions van d'una pàgina a una altra, i comuniquen aquestes dues, i les dades que aquestes poden contenir.

El servidor d'aplicacions corporatiu BEA Weblogic 9.2 és el producte estrella de l'empresa BEA Systems amb el que es refereix a servidors destinats a web, dintre l'oferta de servidors de serveis de tipus SOAP i d'aplicacions corporatius sobre plataformes mòbils.

A més, per al servei de motor de base de dades, s'ha escollit el software SUN Oracle8i. Oracle és una referència en el món de les bases de dades, gràcies a una gran robustesa, adaptabilitat i escal·labilitat. Està composat per diverses aplicacions que permeten una gestió precisa, com ara SQL*Plus i Oracle DBA. A més, té una estructura molt clara de fitxers, basada en el repartiment de les dades i configuracions en diverses parts. Les possibilitats que ofereix son molt àmplies, van des de un servidor local de base de dades, a oferir-ne molts serveis, per a diversos clients, i cadascun amb una sèrie de permisos determinats i configurats. A més, està disponible tant per sistemes Windows, Unix / Linux i sistemes propis de Sun Solaris. Gràcies a una gran compatibilitat amb estàndard com ara ODBC o SQL, permet una interoperabilitat amb una gran varietat de servidors i aplicacions.

La unió d'aquests dos servidors és possible gràcies a la integració que permet BEA Weblogic amb Oracle, a través d'un driver JDBC per Oracle que està incorporat al Workshop for Weblogic. D'aquesta manera, quan hem creat el domini per encapsular el servidor Weblogic, hem activat un nom JNDI que apunti qualsevol connexió JDBC al servidor Oracle que hem preparat. Aquest està configurat per oferir el servei a l'adreça localhost:1521, mentre es proporciona el nom d'usuari i password. D'aquesta manera, Weblogic llença les comandes SQL contra el servidor Oracle, i aquest respon, sense haver d'implementar la connexió JDBC, a través de la inicialització de les classes. Tot aquest procés s'implementa en una subcapa en la qual el programador no té per què accedir-hi.

El primer argument d'aquesta elecció és perquè es la **solució** més **equilibrada**. El servidor d'aplicacions BEA Weblogic és una solució elaborada per un grup de professionals especialitzats en l'àmbit corporatiu, els quals basen el seu treball en un gran servidor d'aplicacions Java, l'Apache Tomcat. Donat aquest caire corporatiu, el servidor d'aplicacions Weblogic respon en molts àmbits, que van des de la creació de portals destinats a la imatge públic a Internet, al desenvolupament d'aplicacions web distribuïdes sobre plataformes mòbils. En referència al servidor de bases de dades, s'ha de destacar que aquest permet una total configurabilitat, una gran compatibilitat i disposa d'un excel·lent suport en funcions de programació

El segon argument és la integració de diferents components que permeten un **desenvolupament net i senzill** d'aplicacions de diversa complexitat, el que el fa apte per albergar aplicacions de àmbit Extranet i Intranet. Els components incorporats representen les tecnologies web més modernes i robustes a la vegada, que permeten la utilització de algorismes simplificats i realment potents.

**BEA Systems Weblogic Server 9.2** és una aplicació de servidor escalable, preparada per Java Two Enterprise Edition (J2EE). La infraestructura de Weblogic Server suporta el desplegament de diferents tipus d'aplicacions distribuïdes i és la solució ideal per a la construcció d'aplicacions basades en Arquitectures Orientades a Serveis (SOA). SOA és una metodologia de disseny dirigida a maximitzar la reutilització entre serveis de les aplicacions.

La implementació completa de la certificació Sun J2EE 1.4 de Weblogic server proporciona un conjunt de API's () estàndard per crear aplicacions Java distribuïdes que són capaces d'accedir a una àmplia varietat de serveis, com bases de dades, serveis de missatgeria, i connexions a serveis empresarials externs.

A més de l'especificació de J2EE, Weblogic Server permet a les empreses desplegar aplicacions de caràcter crític en un entorn robust, segur, escal·lable i d'una alta disponibilitat. Aquestes característiques donen a les empreses la possibilitat de configurar instàncies de clústers de servidors Weblogic per poder distribuir la càrrega de treball i proporcionar una capacitat extra en casos de fallida de hardware o d'altres motius. Les noves ferramentes de diagnòstic permeten als administradors de sistemes monitoritzar i ajustar el rendiment de les aplicacions desplegades i el propi entorn del servidor Weblogic. També es possible configurar el propi servidor Weblogic per monitoritzar i ajustar el throughput de l'aplicació, sense la intervenció de cap operari. Les característiques de seguretat extensibles, protegeixen l'accés als serveis, mantenen segures les dades empresarials i eviten atacs maliciosos.

BEA Weblogic Server s'ofereix amb un entorn de desenvolupament anomenat BEA Workshop for Weblogic. Aquesta solució integra a més el servidor Weblogic. Si analitzem aquesta solució veiem que es tracta d'un servidor Apache Tomcat, amb una implementació paral·lela de Apache Struts. Amb tot això li afegim una extensió de les funcionalitats, orientades cap a la lògica empresarial, mitjançant una màquina virtual de Java desenvolupada anomenada JRockIt. Per aprofitar la suma d'aquests serveis, es proporciona un IDE Eclipse preconfigurat per al servidor que s'acompanya. D'aquesta manera, es poden desenvolupar projectes de gran envergadura partint d'una senzilla implementació. A partir d'aquí només s'ha d'implementar la lògica d'acord amb l'entorn de desenvolupament. Els models de programació de Weblogic Server : **Aplicació web** : proporciona el mecanisme bàsic J2EE per al desplegament de pàgines Web dinàmiques, basades en estàndards J2EE sobre Servlets i Java Server Pages (JSP). Les aplicacions Web també són usades per servir continguts estàtics com HTML i fitxers d'imatge. **Serveis Web** : ofereixen un conjunt compartit de funcions que són disponibles per altres sistemes d'una xarxa, i poden ser usades com a component d'aplicacions distribuïdes basades en Web. **XML** : Les capacitats XML incloent intercanvi de dades, permeten guardar presentacions de dades, independents del seu contingut. **Java Messaging Service (JMS)** : permeten a les aplicacions comunicar-se les unes amb les altres, a través del intercanvi de missatges. Un missatge és una petició, un informe o/i un esdeveniment que conté informació necessària per coordinar la comunicació entre aplicacions. **Java DataBase Connectivity (JDBC)** : proporcionen accessos a recursos DBMS a través de pools. **Adaptadors de Recursos** : permeten connectivitats a sistemes empresarials inferiors o externs. **Enterprise JavaBeans (EJB)** : proporcionen objectes Java per encapsular dades i lògica empresarial. **Remote Method Invocation (RMI)** : és un estàndard Java per a la computació d'objectes distribuïda, permeten a les aplicacions invocar mètodes en objectes remots localment. **API's de seguretat** : permeten integrar autenticació i autorització en la nova aplicació J2EE. També és possible usar les API de implementació de seguretat, per crear les característiques de seguretat ajustades a les necessitats. **Connectivitat de Weblogic amb** Tuxedo **(WTC)** : proporciona interoperabilitat entre el servidor BEA Weblogic i els serveis BEA Tuxedo. WTC permet als clients de Weblogic accedir a serveis de BEA Tuxedo, i al revés, als clients de BEA Tuxedo invocar els EJB de BEA Weblogic en resposta a les seves peticions.

**Dominis de Weblogic Server.** Un domini és la unitat bàsica d'administració per a instàncies de Weblogic Server. Un domini consta de una o més instàncies de Weblogic Server ( i els seus recursos associats) que es gestionen amb un sol servidor d'administració. Es poden definir múltiples dominis basats en diferents responsabilitat d'administració del sistema, límits de l'aplicatiu, o posició geogràfica dels servidor. Per el contrari, es poden utilitzar un sol domini per a centralitzar totes les activitats d'administració de Weblogic Server.

Cada configuració de domini està guardada en un fitxer de configuració separat anomenat config.xml, que està guardat en el servidor d'administració amb altres fitxers com ara logs i fitxers de seguretat. Quan s'usa el servidor d'administració per efectuar una tasca de configuració, els canvis que es fan s'apliquen només per al domini gestionat pel servidor d'administració. Per gestionar un altre domini, s'utilitza el servidor d'administració del domini. Per aquesta raó, les instàncies del servidor, aplicacions, i els recursos en un domini deuria se tractar com a independent de servidors, aplicacions i recursos de un altre domini. No es pot efectuar tasques de configuració o desplegament en múltiples dominis al mateix temps.

**Apache Beehive** és un framework per a Aplicacions Java que permet el desenvolupament d'aplicacions basades en Java EE d'una manera més ràpida i senzilla. Fa ús de diversos projectes Open Source de la Apache Software Foundation com ara XMLBeans. Aquest ha influencia les últimes innovacions en Java 4 que inclou JSR-175 que és una eina per camp d'anotació, mètodes i classes les quals poden ser tractades de modes especials per les eines en temps d'execució. Es construeix en el framework desenvolupat per BEA Systems Weblogic per a la seva sèrie 8.1. BEA va decidir més tard donar el codi a Apache, ja que permetia a una àmplia audiència tenir la oportunitat d'usar Beehive. Beehive es composa de : **Flux de pàgines Netui.** És un framework d'aplicacions construït sobre Apache Struts que simplifica el treball i l'actualització amb els diversos fitxers de configuració de Struts. **Controls.** Aquest és el cor del Framework Beehive. Un control pot ser definit com un programa que pot ser utilitzat pel programador per aconseguir accés ràpidament als recursos de nivell empresarial com poden ser el EJB (Enterprise Java Beans), serveis web, etc. Per exemple, considerant l'accés a un EJB. Aquest implicar molt codi típic, com per exemple, per accedir a la interfície principal, llavors creant/trobant un EJB utilitzant un mètode de cerca i llavors accedint a els mètodes remots del Bean. L'ús del control ho simplifica degut a que el EJB genera aquest codi típic pel programador, que es pot concentrar en la lògica del negoci que preocupant-se dels detalls interns de la tecnologia Java EE. Si es programador està qualificat, trobarà això molt útil, degut a que es podrà concentrat en coses més útil, per exemple, construint un Facade cap a un conjunt complex de API's de l'aplicació. En essència, un control a un EJB assegura que el programador pot simplement utilitzar el control i cridar qualsevol mètode del EJB, usant-lo de la mateixa manera que qualsevol altra classe Java. Els controls venen amb un assistent de conjunt de controls estàndard : EJB Control, Webservice Control, Database Control i JMS Control. Es poden desenvolupar controls personalitzats que a la vegada poden fer ús de controls ja construïts. **Serveis Web.** Aquest és el tercer component de Beehive i permet al desenvolupador crear serveis web ràpidament utilitzant metadades/anotacions. En essència, usant les metadades/anotacions es poden crear serveis web complexes utilitzant funcions com conversació, estat,etc., de forma ràpida i neta, ja que totes les dades estan en un únic fitxer, i fa que sigui més senzill de depurar i mantenir. Utilitzant aquesta aproximació, qualsevol classe Java pot ser convertida en servei web simplement per afegir anotacions dintre els fitxers de codi Java. Està basat en JSR-181 que es basa alhora en JSR-175.

**Apache Struts** és un framework d'aplicacions web de codi lliure per a desenvolupar aplicacions Web sobre Java EE. Utilitza  i amplia la API de Servlets Java per animar als programadors a adoptar una arquitectura MVC. Va ser creat originalment per Craig McClanahan i donat al projecte Apache Foundation al maig del 2000. Anteriorment situat sota el projecte Apache Jakarta i conegut com a Jakarta Struts, ha esdevingut un projecte Apache de primer nivell el 2005. En una aplicació web estàndard de Java EE, el client normalment és qui envia  informació al servidor a través de un formulari web. La informació es tractada per un Servlet Java que la processa, interactua amb la base de dades i produeix una resposta que està fa amb format HTML o bé és transmesa a una JSP (Java Server Pages) que mescla HTML i codi Java per aconseguir un resultat similar. Les dues aproximacions son normalment considerades inadequades per a projectes grans perquè es mescla lògica de l'aplicació amb la de presentació i dificulta el manteniment. L'objectiu de Struts es separar de forma neta el model (lògica d'aplicació que interactua amb la base de dades) de la vista (pàgines HTML presentades al client) i el controlador ( instància que comunica vista i model). Struts proporciona al controlador (un Servlet conegut a ActionServlet) i facilita l'escriptura de plantilles per a la capa de presentació . El programador de l'aplicació Web és responsable de escriure el codi del

model, i de crear un fitxer  de configuració central anomenat struts-config.xml que uneix models, vista i controlador. Les peticions del client son enviades al client en forma de acció definides en el fitxer de configuració; si el controlador rep una petició que crida la corresponent classe Action que interacciona amb el codi específic model de l'aplicació. El codi model retorna un "Action Forward", una String conta al controlador quina pàgina s'ha enviat al client. La informació viatja entre model i vista en forma de JavaBeans especials. Una poderosa llibreria de etiquetes personalitzades permet llegir i escriure el contingut d'aquests beans des de la capa de presentació sense la necessitat de qualsevol codi Java embegut.
Struts també suporta internacionalització i18N, proporcionant facilitats per a la validació de dades enviades per formularis web, i inclou un mecanisme de plantilles, anomenat "Tile" que (per instancia) permet a la capa de presentació estar composta de components de capçalera, peu de pàgina i contingut diferents.

**L'entorn de desenvolupament Workshop for Weblogic.** L'entorn de desenvolupament BEA Workshop for Weblogic es un aplicatiu basat en el IDE Eclipse. Proporciona un espai basat en perspectives, configuracions de pantalla que permet visualitzar diferents conjunts de finestres. La principal es conforma amb el navegador de solucions, el resum del flux de pàgines, i la finestra de codi pròpiament.

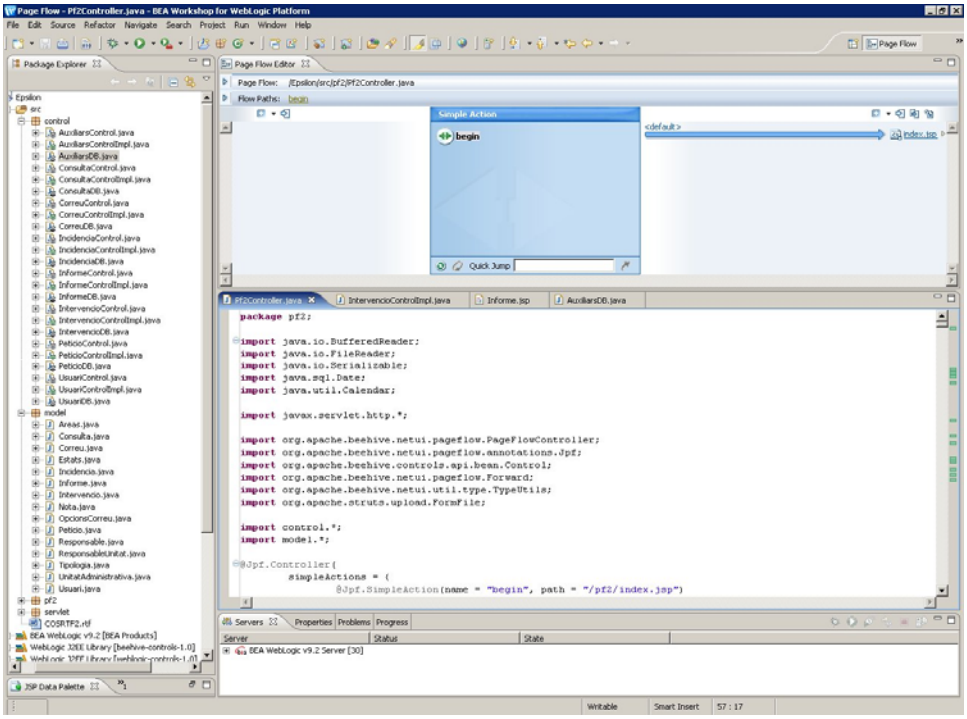



**Diagrama 6. Captura de pantalla del IDE BEA Workshop.**

Aquest IDE a més, permet gestionar el servidor Weblogic i activar el mode debug, que permet buscar errades dintre l'aplicatiu mitjançant una perspectiva especial. BEA Workshop for Weblogic, al utilitzar la base de Eclipse, permet la utilització de plugins, de manera que la seva funcionalitat és altament extensible. Les característiques més destacades d'aquest Entorn de Desenvolupament :

- Mode de debug complet, amb trace i consulta de variables i de fills del procés.
- Anàlisis de errors de codificació amb temps real.
- Control de l'estat de compilació dels fitxers de codi.
- Representació gràfica del(s) flux(es) de navegació.
- Cerca de mètodes, pàgines i classes.

- Identificació de comandes, paraules reservades i etiquetes en temps real.
- Control del servidor Weblogic, i el domini associat.
- Editor JSP no WYSIWYG, el que permet una millor codificació.
- La incorporació de diversos de projectes Java simple a Portals Web Java.

**SUN Oracle 8.1i**, en general, es tracta d'una solució molt completa i complexa de gestió de dades. Està composta de un servidor en xarxa de base de dades, servidor d'aplicacions integrat, aplicacions de gestió de les bases de dades, de manteniment, de programació, etc.

S'utilitza bàsicament com a servei de dades, a utilitzar amb aplicacions on els usuaris generen diverses peticions, de manera que els diversos clients poden accedir i modificar les dades en el mateix moment.

**Instàncies**. Una instància Oracle és un servei de bases de dades que ofereix el sistema Oracle. De manera que podem tenir diversos serveis a la vegada funcionant en un servidor. Solen anar diferenciats per les taules als que tenen accés, tot i que, aquesta explicació no és del tot exacta. La instància consta de diverses parts, aquestes en conjunt, possibiliten l'accés a les dades, a més de la seva estabilitat i accessibilitat :

A) Cinc processos: Monitor del sistema, Monitor dels processos, Escriptor de la Base de dades, Punt de control i escriptor del registre.
B) Fitxers de dades que contenen les taules i altres objectes de dades, fitxers de control que contenen la informació de la configuració, registre de *redo* per al processament de les transaccions i els fitxers de suport per a propòsits de recuperació.

C) Fitxers de configuració que contenen els atributs de les instàncies i informació de la seguretat externa.

**Model de Memòria.** El model de memòria de Oracle consta de:

A) el SGA, que conté el codi del executable del programa (min 8 MB).

B) el fons compartit, que està separat en el cache de la biblioteca de dades, i el cache del diccionari de dades. Les consultes optimitzades són guardades aquí.

C) El búffer del registre de redo, que és on les transaccions són guardades abans que siguin escrites sobre els registres.

D) El búffer del cache de la BBDD, que és on les operacions romanen en espera mentre són escrites als fitxers de dades.

El servidor de dades Oracle funciona com múltiples processos dins del sistema operatiu. El nombre d'usuaris que estan connectats a la base de dades es reflexa amb el nombre de processos gestionats pel SO. Cada usuari que es connecta requereix 1 MB de memòria del servidor.

**Procés de transaccions**. Les transaccions són escrites al búffer del registre de redo, on van avançant consecutivament als registres, al búffer del fitxer de dades i finalment, al fitxer de dades. Quan ocorre un rollback, les imatges dels blocs de dades son descartats del búffer de registre de redo; tal que el bloc anterior és mantingut en el blocs de segment del rollback. Les transaccions compromeses són promocionades als registres de arxivament. Aquests són usats per restaurar les dades quan hi ha una fallida de hardware.

Una operació de punt de control descarta tots els blocs de memòria que siguin actualitzats (i compromesos) dels búffers de registre i del fons de búffer de la base de dades. Remarcar que

l'enregistrament de les transaccions és opcional per les bases d'una taula, i l'enregistrament de arxivament també és opcional. Durant una transacció d'actualització, el bloqueig de registre proporciona prevenció contra col·lisions de blocs de dades. Accés a la imatge anterior dels registre s'ha fet disponible durant aquest temps, que redueix el temps de contenció. Aquest és un mecanisme de bloqueig de registre patentat.

**Procediment de Backup i recuperació.** Abans de Oracle 8, la única manera d'obtenir una còpia de seguretat era mitjançant un backup en "gelat". Això involucrava aturar la instància de Oracle, fent la còpia de seguretat les fitxers de dades i reiniciant. Oracle 8 té el paquet anomenat Gestor de Recuperació, que facilita salvaguardar les dades mentre aquestes estan on-line. La recuperació s'aconsegueix per restaurar els fitxers de dades, i verificant que els fitxers de control estiguin sincronitzats apropiadament.

**Posta a punt de la Seguretat i dels Comptes d'usuari.** Oracle es tramitat amb diverses comptes preestablertes : SYSTEM, INTERNAL i SYS. Es requereix una autenticació del Sistema Operatiu en ordre per crear un compte que tindrà privilegis similars. Després de que el compte sigui creat, es permet l'accés a les taules dins dels esquemes segons les necessitats del compte.

**Creació de les Bases de Dades.** Les bases de dades són inicialitzades amb la comanda "CREATE DATABASE". En la majoria dels casos (99.9%) el nom de la base de dades és el mateix que el nom de la instància, i només hi ha una base de dades per instància.

Dins d'una instància de Oracle, els esquemes són creats que continguin les taules per a una aplicació o utilitat. Les taules són referenciades mitjançant NOM_ESQUEMA.NOM_TAULA. Cada usuari és assignat a un esquema per defecte sobre la creació; aquest nom de l'esquema és el mateix que el nom d'usuari. En línia que estiguin referenciades sense el nom de l'esquema, aquestes han de ser "propietat" de l'usuari, o bé estar dins de l'esquema "system". Els "Sinònims" poden de creats per saltar el requeriment del prefix. Normalment, un usuari o login de tipus "internal" o "system" serà usat per accedir a la base de dades. Una instància típica de Oracle tindrà 12 fitxers de dades, 6 membres de registre per *redo*, 6 fitxers d'arxivament i 4 fitxers de control, tots estos a través de diversos sistemes secundaris del disc.

**Tipus de Dades.** El tipus de dades suportats inclouen *number, char, varchar2, date, long raw, clob* i *blob*. El tipus de dades *blob* i *clob* estan implementats mitjançant punters, sense la estructura de registre físic de disc. Els continguts dels camps estan guardats en blocs dedicats. Per tant, cada camp blob o clob requereix com a mínim 2KB d'espai (depenent de la mida de bloc de la base de dades). Els tipus long raw estan guardats de manera contínua, i estan desaconsellats.
Per dades string, el tipus varchar2 pot ser utilitzat per longituds fins 2000 caràcters. El tipus clob pot ser utilitzat per camps de dades més llargs.
Els camps Date estan representats com a nombre de dies, utilitzant fraccions decimals, amb precisió de minuts). Es pot obtenir més granul·laritat comprant un mòdul separat.
Les seqüències proporcionen un tipus de autoincrement per a una columna ID, seleccionant "sequence_name.NEXTVAL" automàticament incrementa la seqüència i li retorna el nou valor.

**Conceptes d'emmagatzematge.** Les taules estan guardades en tablespaces, un espai de taula es fet de un o més fitxers de dades. Sinó es possible de usar dispositius RAW dins de Oracle, no és aconsellat. Els fitxers de control, el segments de "rollback", i els registres de "redo" estan tots establerts en fitxers separats dins del sistema operatiu.

**PL-SQL.** PL-SQL es un llenguatge de programació robust que permeten crear procediments establerts. Els procediments estan guardats en un format compilat, el que permet una execució de codi més ràpida. El cursors estan suportats en processaments de fila en fila. Les matrius estan suportades, mitjançant (el tipus de dades de taula). Des de que els procediments PL-SQL no poden retornar un Resultset, per retornar només les files des d'una crida d'una aplicació requereix una

implementació de matrius com a variables de sortida. Una poderosa funcionalitat de PL-SQL és la possibilitat de crear funcions personalitzades que poden ser utilitzades en sentencies SQL.

**Rendiment i escalabilitat.** Oracle sempre s'ha conegut per la seva velocitat i rendiment. Oracle 8 dóna suport per unes 15.000 connexions actives d'usuari. L'esquema de bloqueig de registres patentat l'ha fet un candidat atractiu per aplicacions embegudes comercialitzades per PeopleSoft i similars. La habilitat per tornar els registres de transacció on i off permetrà a Oracle superar als competidors en anàlisis de funcionament.

**Preu i suport.** El preu per llicència és relativament alt, comparat amb altres subministradors. El suport s'aconsegueix obrint els casos "TAR" amb l'equip de suport. La resposta es dona normalment dins de 48 hores.

**Eines de gestió i desenvolupament (per Windows).** SQL Plus es la eina de consulta interactiva utilitzada amb Oracle, és útil per introduir consultes i procediments establers. Les capacitats per generar informes complexes estan disponibles per a aquells que arenguin les extensions de SQL Plus. Oracle Enterprise es proporciona amb el gestor d'empreses, un frontend ple de funcionalitats de les capacitats intrínseques de Oracle. Les àrees principals del sistema (seguretat, emmagatzematge i esquemes) estan gestionar per aplicacions completament separades, que a vegades poden ser enyoroses. El producte "millor de la classe" en la seva categoria es DB Artisan d'Embarcadero Technologies. Les altres eines de gestió del sistema Oracle (com la GUI gestor per a Context) tenen moltes raons per ser desitjades. Oracle s'ha guanyat una reputació de les eines de desenvolupament de pobra qualitat amb el llançament de Oracle Forms per Windows. Posteriorment anomenat Developer 2000, s'està guanyant l'acceptació dins dels clients de Oracle.